WAP-Services für die Hochschule der Medien: Erreichbarkeitsauskunft und Verzeichnisdienst

Diplomarbeit
im Fach Informationsnetze, Kommunikationstechnik und Netzwerkmanagement
Studiengang Informationsmanagement
der
Fachhochschule Stuttgart –
Hochschule der Medien

Susanne Martina Klischat

Erstprüfer: Prof. Dr. Wolf-Fritz Riekert
Zweitprüfer: Prof. Dr. Christian Rathke

Bearbeitungszeitraum: 01. August bis 09. November 2001

Stuttgart, November 2001

Kurzfassung

Diese Diplomarbeit beschreibt die Konzeption und Realisierung von WAP-Services für die Hochschule der Medien. Konkret geht es um eine Erreichbarkeitsauskunft mit parallelem Verzeichnisdienst.

Zu Anfang der Arbeit wird ein kurzer Überblick über das Wireless Application Protocol und zugehörige Komponenten gegeben. Vor- und Nachteile, die sich aus dem WAP als Grundlage ergeben, werden angesprochen.

Der Benutzer kann über die WAP-Services Auskünfte zur Erreichbarkeit einzelner Dozenten einholen. Ihm wird angezeigt, ob ein Dozent zu einem bestimmten Zeitpunkt gerade eine Lehrveranstaltung hält oder abwesend ist. Zusätzlich werden dem Benutzer Kontaktdaten der gesuchten Person präsentiert. Durch das Drücken weniger Tasten kann er eine angezeigte Telefonnummer sofort anrufen oder abspeichern. Diese Funktionalität wurde über Wireless Telephony Application Interface (WTAI) Bibliotheken realisiert.

Die Datengrundlage für die Erreichbarkeitsauskunft und den Verzeichnisdienst bildet die bereits bestehende Stundenplan-Datenbank, ergänzt um eine weitere Datenbank.

Die WAP-Services sind über den Lehre-Server der HdM verfügbar. Hierbei handelt es sich um ein LAMP-System (Linux - Apache - MySQL - PHP). Die Programmierung der WAP-Services erfolgte in WML, WMLScript, PHP und SQL.

Schlagwörter: Wireless Application Protocol, WAP, Erreichbarkeit, Verzeichnisdienst, Mobile Information, LAMP, Handy, Mobiltelefon

Abstract

This thesis describes the conception and realization of a WAP Service for the School of Media in Stuttgart. It begins with a short description of the Wireless Application Protocol (WAP) and its components. Advantages and disadvantages resulting of the WAP are discussed.

The implemented WAP services allow users to request information about a professor's availability through a mobile phone. It is displayed whether the professor is lecturing or traveling at a certain point in time. In addition, the users are presented with contact data of the desired person. With only a few keystrokes, they can have their mobile devices either dial or save any of the displayed phone numbers. This service is realized using the Wireless Telephony Application Interface (WTAI) Library.

The already existing timetable database and another recently created database with additional information form the basis for availability and contact data.

All these WAP Services are available on the 'Virtual University Server' of the School of Media (wap://v.hdm-stuttgart.de). This server is a LAMP system (i.e., it makes use of the Linux, Apache, mySQL, and PHP technologies). The WAP Services are programmed in the WML, WMLScript, PHP and SQL computer languages.

Keywords: Availability, Wireless Application Protocol, WAP, Directory Service, LAMP, Mobile Information, Cellular phone, Cell phone

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ASP	Active Server Pages
CSD	Circuit Switched Data
CSS	Cascading Style Sheets
DTD	Document Type Definition
DTMF	Dual Tone Multi-Frequency
EDGE	Evolved Data for GSM Evolution
FB	Fachbereich
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communication
HBI	Hochschule für Bibliotheks- und Informationswesen
HdM	Hochschule der Medien
HDM	Hochschule für Druck und Medien
HSCSD	High Speed Circuit Switched Data
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
LAMP	Linux-Apache-mySQL-PHP
LDAP	Lightweight Directory Access Protocol
LV	Lehrveranstaltung
MESZ	Mitteleuropäische Sommerzeit
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messaging Service
MSISDN	Mobile Station International Subscriber Directory Number
OSI	Open Systems Interconnection
PC	Personal Computer
PDA	Personal Digital Assistant
PHP	PHP: Hypertext Preprocessor
SMS	Short Message Service

SQL	Structured Query Language
TCP	Transmission Control Protocol
UDP	Universal Datagram Protocol
UMTS	Universal Mobil Telecommunications System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAE	Wireless Application Environment
WAP	Wireless Application Protocol
WBMP	Wireless Bitmap
WBXML	WAP Binary XML Content Format
WDP	Wireless Datagram Protocol
WSP	Wireless Session Protocol
WTLS	Wireless Transport Layer Security
WTP	Wireless Transport Protocol
WML	Wireless Markup Language
WML1	Wireless Markup Language Version 1.x
WML2	Wireless Markup Language Version 2.0
WMLScript	Wireless Markup Language Script
WP-HTTP	Wireless Profiled Hypertext Transfer Protocol
WP-TCP	Wireless Profiled Transmission Control Protocol
WTA	Wireless Telephony Application
WTAI	Wireless Telephony Application Interface
WWW	World Wide Web
XHTML	Extended Hypertext Markup Language
XML	Extended Markup Language

1 Überblick

Wozu braucht eine Hochschule WAP-Services? Ist es gerade in Mode, gehört es einfach dazu, im mobilen Internet präsent zu sein? Für die im Rahmen dieser Diplomarbeit entwickelten WAP-Services trifft das sicherlich nicht zu.

Ich will den Nutzen der WAP-Services anhand dreier Szenarien veranschaulichen:

- Es ist Freitag nachmittag. Ein Student[1] ist sich nicht sicher, ob der außerplanmäßige Workshop für dieses Wochenende geplant war oder erst für eins der kommenden. Er erreicht gerade keinen seiner Kommilitonen, den er fragen könnte. Trotzdem würde er es gerne so schnell wie möglich erfahren, weil er eigentlich über das Wochenende nach Hause fahren wollte.
- Eine Professorin plant mit einem anderen Professor und mehreren Studenten ein Projekt. Eigentlich wäre heute Abend eine Besprechung geplant gewesen, ihr ist aber etwas sehr dringendes dazwischen gekommen. Sie will den anderen Projektteilnehmern bescheid geben, daß sie nicht kommen wird. Gerade als sie ihren Kollegen anrufen will, merkt sie, daß sie ihr Adressbuch im Büro liegengelassen hat. Um diese Uhrzeit ist die Telefonzentrale der Hochschule nicht mehr besetzt und auswendig weiß sie die Telefonnummer nicht mehr.
- Ein Student würde gerne Lehrveranstaltungen eines anderen Fachbereichs besuchen. Vorher will er allerdings den zuständigen Professor noch fragen, ob in der Veranstaltung noch ein Platz für ihn frei ist. Die Vorlesung findet am Standort Wolframstraße statt, dort hat auch der Professor sein Büro. Der Student ist gerade in der Stuttgarter Innenstadt und überlegt sich, ob er einen Abstecher in die Wolframstraße machen soll. Allerdings ist es bereits Spätnachmittag und er ist sich nicht sicher, ob der Professor noch da sein wird. Wäre er jetzt an der Hochschule, könnte er im Internet im Vorlesungsverzeichnis nachschlagen, wann der Professor seine Vorlesungen hält. Aber mitten auf der Königsstraße? Er könnte ja versuchen, den Professor kurz anzurufen, aber welche Telefonnummer hat sein Büro? Oder sollte er doch lieber eine E-Mail schicken? Die Adresse würde er ebenfalls irgendwo auf den Webseiten der Hochschule finden.

Alle drei Personen suchen eine bestimmte Information, sind aber durch ihre Umstände daran gehindert, sie zu bekommen. Beispielsweise, weil Sie gerade keinen Computer mit Internetzugang in der Nähe haben, um auf den Hochschulwebseiten im Vorlesungs-

[1] Aufgrund der besseren Lesbarkeit verwende ich überwiegend die männliche Form (*Student, Professor, Benutzer,...*), hierbei ist selbstverständlich die weibliche Form mitgemeint (*Studentin, Professorin, Benutzerin...*). Für *Benutzer* weiche ich daher teilweise auch auf den geschlechtsneutralen englischen Ausdruck *User* aus.

verzeichnis nachzuschlagen oder weil die Telefonzentrale und das Sekretariat nicht mehr besetzt sind, aber eigentlich die gewünschte Auskunft hätten geben können.

Hier kommt einer der großen Vorteile von WAP zum Tragen: Mobiltelefon sind klein und man hat sie eigentlich fast immer und überall dabei. Mit einem WAP-fähigen Mobiltelefon könnten die drei oben genannten Personen sofort die Information bekommen, die sie suchen: Die WAP-Services liefern Informationen über die Erreichbarkeit eines Dozenten, beispielsweise ob er im Moment eine Lehrveranstaltung hält. Zusätzlich können für diesen Dozenten Telefonnummer, E-Mail-Adresse oder sonstige Adress- und Kontaktdaten abgerufen und abgespeichert werden.

Ein Argument der WAP-Gegner ist natürlich, daß die Bedienung einer WAP-Anwendung sehr unkomfortabel ist: kleines Telefondisplay, kleine Tasten, langsame Übertragungsgeschwindigkeiten. Es nicht unbedingt ein Vergnügen, mit WAP zu „surfen“, wie man es an einem PC machen würde. Letztlich geht es hier aber um Informationen mit einem konkreten Nutzen. Eine gut durchdachte WAP-Anwendung wird dem Benutzer die Möglichkeit geben, so schnell und effizient wie möglich an Informationen zu kommen.

2 Zielsetzung

Mit der Erreichbarkeitsauskunft und dem Verzeichnisdienst soll eine einfach zu bedienende, schnelle Auskunft über WAP ermöglicht werden. Der Benutzer sollte so schnell wie möglich seine Information bekommen, ohne viele Angaben machen zu müssen: „Three clicks and you're out."[2] (Taylor/Hosking/ Brazier 2000)

Wie der Name Erreichbarkeitsauskunft schon sagt, liefert sie Informationen zur Erreichbarkeit einer Person. Erreichbarkeit bedeutet in diesem Zusammenhang „an der Hochschule erreichbar". Der Benutzer erhält jedoch keine „Ja/Nein"-Antwort, sondern Informationen, anhand derer er entscheiden kann, wie er weiter verfährt, ob er die Person aufsucht, oder ob er es lieber zu einem anderen Zeitpunkt noch einmal probiert.

Da die Datengrundlage der Erreichbarkeitsauskunft die Stundenplan-Datenbank bildet, ist die Anwesenheit bzw. Erreichbarkeit an der Hochschule geknüpft an die Lehrveranstaltungen eines Dozenten. Es besteht keine andere Möglichkeit, seine Anwesenheit mit bereits bestehenden Datenbanken o. ä. zu ermitteln (wie die Stempeluhr in einem Industriebetrieb). Würde man das Konzept der Erreichbarkeitsauskunft vom Hochschulbetrieb in die Wirtschaft übertragen, sähe die Datengrundlage (Stempeluhr, gemeinsame Terminplanung) und damit die Definition der Anwesenheit natürlich anders aus.

Beispielhafte Auskünfte an einen Benutzer könnten aussehen wie folgt:

- „Dr. XY hält im Moment die Lehrveranstaltung ‚Information und Gesellschaft' in Raum 012. Die Lehrveranstaltung endet voraussichtlich um 15:00 Uhr."
- „Die letzte Lehrveranstaltung von YZ endet um 18:15 Uhr. Es sind keine weiteren Vorlesungen für diesen Tag eingetragen."
- „Um 13:45 endet die letzte Lehrveranstaltung von ZZ, um 17:00 beginnt die nächste Veranstaltung in Raum 400."

Die Auskunft an sich soll in einem natürlichsprachigen Satz erfolgen, um das Lesen zu erleichtern. Außerdem sollten sie die 160-Zeichen-Grenze von Kurznachrichten nicht überschreiten, da der Benutzer es nicht gewohnt ist, Texte, die länger sind,. auf dem Telefondisplay zu lesen.

Der Verzeichnisdienst und die Erreichbarkeitsauskunft sind integrierte Systeme. Sobald Erreichbarkeitsinformationen ausgegeben werden, soll der Benutzer auch die Möglichkeit haben, Kontaktdaten zur gewünschten Person abzurufen (z. B. Telefonnummer, Anschrift, E-Mail-Adresse). Verzeichnisdienst und Erreichbarkeitsauskunft arbeiten unter derselben Oberfläche, es können aber auch Kontaktdaten abgerufen werden, ohne eine Erreichbarkeitsauskunft einzuholen und umgekehrt.

Die Kontaktdaten sollen mit entsprechenden Endgeräten als Visitenkarte (vCard) zur Speicherung bereitstehen. Telefonnummern sollen über WTAI-Funktionen ebenfalls zum Anruf oder zur Speicherung bereitstehen.

Eine Anmerkung noch, um Verständnisproblemen vorzubeugen: Beim Verzeichnisdienst handelt es sich nicht um einen LDAP-Verzeichnisdienst (z. B. unternehmensweite E-Mail-Adressbücher, die über das Lightweight Directory Access Protocol angesprochen werden).

[2] In Anlehnung an den Paragraphen 667 des kalifornischen Strafgesetzbuchs: „Three strikes and you're out.": Nach drei Straftaten droht eine längere Gefängnisstrafe.

3 Status

Eine ähnliche Funktionalität wie die der Erreichbarkeitsauskunft und des Verzeichnisdienstes, lässt sich mit den bisherigen Systemen der HdM, Fachbereich 3 nicht erzielen. Sie lässt sich höchstens durch die Kombination mehrerer Informationsressourcen imitieren:

- Es gibt das Dozenten- und Raumauskunftssystem auf http://www.v.hdm-stuttgart.de/. Hiermit können ein Stundenplan für einen Dozenten bzw. ein Belegungsplan für einen Raum erstellt werden. Die gezielte Suche nach einem Zeitpunkt ist nicht möglich, man kann lediglich die Woche bestimmen, auf die sich der Plan beziehen soll.
  Für einige WAP-fähige Endgeräte gibt es auch HTML-Browser (einige Personal Digital Assistants, Nokia Communicator, Ericsson 380), so daß sie auf diese Seiten zugreifen könnten, was allerdings sehr unkomfortabel ist, da ein ausgegebener Stundenplan als Tabelle mit mehr als zehn Zeilen und sieben Spalten angezeigt wird. Außerdem werden Zusatzinformationen (wie der Titel der Vorlesung) in einem eigenen Frame dargestellt, was Anzeigeprobleme verursachen kann.
- Adressen und Telefonnummern findet man entweder über das gedruckte Vorlesungsverzeichnis oder auf den Webseiten der jeweiligen Professoren.

Die Erreichbarkeitsauskunft und der Verzeichnisdienst sind insofern eine besondere Kombination dieser beiden Aspekte mit dem Mehrwert, von überall zugänglich zu sein (sofern dort ein Mobilfunknetz erreichbar ist).

4 WAP

Das Wireless Application Protocol (WAP) wurde 1998 zum ersten Mal der Öffentlichkeit vorgestellt. Das Ziel war, einen drahtlosen Internet-Zugriff von bestimmten, meist mobilen Endgeräten, wie Mobiltelefonen oder Personal Digital Assistants (PDA) zu ermöglichen.

WAP ist aber nicht zwangsläufig an den Mobilfunk gebunden. Auf über WAP angebotene Inhalte können entsprechende User Agents (wie Simulatoren oder WML-Browser) ebenso über die Internet-Protokolle TCP/IP und HTTP zugreifen. Ein User Agent ist laut WAP Forum definiert wie folgt: „A user agent is any software or device that interprets WML, WMLScript, WTAI or other resources. This may include textual browsers, voice browsers, search engines, etc." (WAP Forum 2001a, S. 8)

Die große Herausforderung bei der Entwicklung von WAP war und ist, den (begrenzten) Fähigkeiten der Endgeräte gerecht zu werden. Einschränkungen, die bei der Nutzung über ein Mobiltelefon als typischer User Agent berücksichtigt werden müssen, sind:

- Kleine Displays (im Durchschnitt etwa 100 Pixel Breite und 50 Pixel Höhe, es sind jedoch auch wesentlich kleinere Displays möglich), meist mit monochromer Darstellung
- Kein sehr leistungsstarker Prozessor
- Geringe Speicherkapazität (z. B. besitzt das Nokia 7110 1,3 KB Speicher, Siemens S/C/M 35 1,5 KB, Ericsson R320 30 KB)
- Keine einheitlichen Schriftarten, und -größen
- Beschränkte Eingabemöglichkeiten, die meistens nur mit einem Finger bedient werden
- zur Zeit noch geringe Bandbreite des Mobilfunknetzes (9.600-28.600 Bit/s)
- Hohe Latenzzeit (Latency) des Funknetzes

Ein Ziel für die Entwicklung des WAP war, die besonderen Eigenschaften der Mobiltelefone und Mobilfunknetze zum Vorteil zu nutzen(vgl. WAP Forum 2001a, S. 12). Zu diesen Eigenschaften zählen unter anderem:

- Mobilität: Durch die inzwischen fast vollständige Netz-Abdeckung (in Deutschland bzw. Europa) kann mit einem Mobiltelefon fast überall kommuniziert werden.
- Kleine Geräte: Mobiltelefone sind inzwischen so klein und leicht, daß sie der Nutzer immer bei sich haben kann (im Gegensatz zu einem tragbaren Computer beispielsweise).

- Ortung: Bedingt durch die Funkzellen-Struktur der Mobilfunknetze ist es möglich den Standort eines Teilnehmers zu bestimmen. Ihm könnten dann standortbezogene Informationen angeboten werden (Location Based Services), beispielsweise Restaurant-Empfehlungen, Informationen zu Übernachtungsmöglichkeiten oder Sehenswürdigkeiten.

4.1 Vorbemerkung

Bevor ich auf die genauen WAP-Versionen und Spezifikationen eingehe, will ich noch etwas Begriffsklärung betreiben. In der Werbung wird WAP meistens in Zusammenhang mit UMTS erwähnt: UMTS als Nachfolger von WAP, UMTS wird WAP überflüssig machen...“ Dies ist jedoch Nonsens, da sich WAP und UMTS in verschiedenen Schichten des (Telefon-)Netzwerks befinden: UMTS bezeichnet eine Mobilfunknetztechnologie, während WAP ein Protokoll ist, daß eben diese Technologie (aber auch eine andere) als Träger benutzen kann. Mit der besseren Übertragungsrate, die UMTS bietet, kann sich WAP weiterentwickeln und neue Funktionen integrieren, die erst durch die schnelleren Verbindungen möglich werden (z. B. der Versand von Multimedia-Nachrichten).

Die wichtigsten Mobilfunknetz-Technologien sind:

- GSM (Global System for Mobile Communication) ist ein Mobilfunknetz der zweiten Generation und ist zur Zeit der Standard für digitale zellulare Mobilfunknetze.
- EDGE (Evolved Data for GSM Evolution) ist eine Übergangslösung bis UMTS funktioniert. EDGE arbeitet mit bestehenden GSM-Netzen, erhöht allerdings die Übertragungsrate. Mit EDGE können Übertragungsraten von bis zu 692.000 Bit/s erzielt werden (allerdings nur mit maximaler Kanalbündelung). Ein Wert von 38400 Bit/s kann ebenso realistisch sein. EDGE erlaubt sowohl paket- als auch leitungsorientierte Verbindungen. Es ist noch nicht ganz sicher, inwiefern EDGE von den Mobilfunknetz-Betreibern überhaupt eingeführt wird.
- UMTS (Universal Mobile Telecommunications Systems) ist ein Mobilfunknetz der dritten Generation. Die ersten Testinstallationen und -verbindungen wurde bereits durchgeführt, allerdings wird es noch eine Weile dauern, bis es flächendeckend verfügbar ist. UMTS bietet neue Frequenzen und wesentlich höhere Datenübertragungsraten, u. a. ist von 384.000 Bit/s bis 2.000.000 Bit/s (2 MBit) die Rede.

Aufbauend auf GSM gibt es drei Trägerdienste, die für WAP von Bedeutung sind:

- CSD (Circuit Switched Data) beschreibt die bisherige Art Daten in einem Mobilfunknetz zu versenden. CSD ist verbindungsorientert und arbeitet mit 9600 Bit/s.
- HSCSD (High Speed Circuit Switched Data) ist ein Nachfolger von CSD. HSCSD arbeitet ebenfalls leitungsvermittelt. Für HSCSD werden mehrere Kanäle (bis zu 8 sind möglich, aber bisher nicht realistisch) mit 9.600 Bit/s bzw. bei verbesserter Fehlerkorrektur 14.400 Bit/s gebündelt. Diese Kanäle werden auf Up- und Down-

link verteilt (sozusagen die Verbindung in das und aus dem Netz), normalerweise stehen dabei dem Benutzer etwa 28.000 Bit/s zur Verfügung.

- GPRS (General Packet Radio Service) bringt ein neues Abrechnungsverfahren mit sich, da es paketorientiert arbeitet und nach Datenmenge abgerechnet wird. Bei GPRS teilen sich bis zu acht Gespräche einen Kanal über Zeitschlitze (time slots) von 577 Mikrosekunden. Die Übertragungsrate berechnet sich aus der Zahl der verfügbaren Slots und deren Übertragungsrate. Zur Zeit ist die typische Übertragungsrate 28.600 Bit/s, maximal wären 171200 Bit/s möglich (aber nicht realistisch).

4.2 WAP-Versionen und Spezifikationen

Die Standardisierung von WAP unterliegt dem unabhängigen Gremium WAP Forum. Es wurde 1997 gegründet von Ericsson, Motorola, Nokia und Unwired Planet (später Phone.com, heute fusioniert mit Software.com zu Openwave Systems). Inzwischen sind im WAP Forum Vertreter fast aller wichtigen Telekommunikations- und Informationstechnologieunternehmen zusammengeschlossen.

4.2.1 WAP 1.x

Die 1998 vorgestellte Version WAP 1.0 definierte die Grundlagen: Wie läuft die Kommunikation zwischen dem User Agent und dem Web-Server bzw. dem dazwischen liegenden Gateway ab. WML und WMLScript waren auch bereits enthalten.

Im Juni 1999 folgte Version 1.1. Die Unterschiede zwischen Version 1.0 und 1.1 betreffen vor allem die WML-Syntax. Alle Tags werden nun klein geschrieben, einige Tags wurden umbenannt oder gestrichen. WAP 1.1 wird von so gut wie allen User Agents verstanden.

Die im November 1999 veröffentlichte Version 1.2, sowie die etwa ein halbes Jahr später vorgestellte Version 1.3, brachten nur noch minimale Änderungen.

4.2.2 WAP 2.0

Im Herbst 2000 wurde sie bereits angekündigt, im August 2001 erschien Version 2.0. WAP 2.0 setzt auf eine höhere Integration bestehender Internet-Protokolle und Standards. Beispielsweise basiert WML2, der Nachfolger der bisherigen Wireless Markup Language, auf XHTML und unterstützt Cascading Style Sheets (CSS) zur Formatierung der WML-Seiten. Außerdem wird das Transmission Control Protocol (TCP) sowie das Hypertext Transfer Protocol (HTTP) unterstützt. Interessant für den Endnutzer sind die Einführung von Multimedia Messaging Service (MMS), d. h. die Möglichkeit verschiedenste Inhalte (z. B. Text, Bild, Film) zu versenden, der Datensynchronisation über das Format SyncML, beispielsweise mit Terminplanungssoftware oder Adressdatenbanken, sowie von standortbezogener Information. Desweiteren werden Farbdisplays, animierte Grafiken und erweiterte Gestaltungsmöglichkeiten für Anzeigemenüs berücksichtigt.

WAP 2.0 ist die Reaktion auf die Mobilfunk-Technologien der 2,5. bzw. 3. Generation (2.5G bzw. 3G), die eine wesentlich höhere Übertragungsrate erlauben (GPRS, HSCSD, EDGE und UMTS).

4.2.3 Wichtige Spezifikationen

Zusätzlich zur WAP-Spezifikation gibt es verschiedene Unterspezifikationen, die weitere wichtige Komponenten definieren.

Unter http://www.wapforum.com/what/technical.htm findet man eine Auflistung aller Spezifikationen. Die wichtigsten sind:

- WAP-210-WAPArch-20010712-a (WAP Forum 2001a) beschreibt die Architektur des Wireless Application Protocols.
- WAP-238-WML-20010626-p[3] (WAP Forum 2001b) beschreibt die Wireless Markup Language.
- WAP-193-WMLScript-20001025-a (WAP Forum 2000a) und WAP-194-WMLScriptLibraries-20000925-a (WAP Forum 2000b) beschreiben die Basisfunktionalität von WMLScript und den zugehörigen Bibliotheken.
- WAP-266-WTA-20010711-p (WAP Forum 2001c) und WAP-268-WTAI-20010715-p (WAP Forum 2001d) beschreiben die Wireless Telephony Application sowie das Wireless Telephony Application Interface.

4.2.3.1 Die WAP-Architektur

Web-Server fungieren oftmals gleichzeitig als WAP-Server. Daher sind die WAP-Architektur und WAP-Protokolle an die Architektur und Protokolle des World Wide Web (WWW) angelehnt.

Beispielsweise kann der Web-Server eines Unternehmens ein HTML-Angebot haben (z. B. Informationen über das Unternehmen, einen Online-Shop), das durch ein WML-Angebot ergänzt wird, um Kunden beispielsweise die Möglichkeit zu geben, sich jederzeit über den Status ihrer Bestellung informieren zu können. Die Adressen der beiden Angebote werden sich nicht einmal sehr unterscheiden, denn WAP benutzt das URL-Schema des World Wide Web: http://v.hdm-stuttgart.de/wap verweist auf WML-Seiten, während http://v.hdm-stuttgart.de auf das HTML-Angebot der Virtuellen Hochschule führt.

Abbildung 1 und Abbildung 2 zeigen World-Wide-Web- und WAP-Modell im Vergleich. WAP benutzt ebenfalls den Anfrage-Antwort-Mechanismus (Request-Response-Mechanism) des WWW. Seit WAP 1.2 ist die Ergänzung durch Push-Funktionen vorgesehen (allerdings sind diese mit den heutigen Endgeräten noch nicht nutzbar).

---

[3] Spezifikationen mit der Endung „-p“ besitzen noch den Status „proposed“ (vorgeschlagen) und werden in nächster Zeit als „approved“ (anerkannt) publiziert.

Push bedeutet, daß der Server Informationen an den Client liefert, ohne daß der Client diese Informationen angefordert hat. Bisher funktioniert der Push von Informationen nur über den Short Message Service (SMS): Nachdem ein Kunde sich entsprechend angemeldet hat, bekommt er Kurznachrichten zugeschickt, die seinem Interessenprofil entsprechen (z. B. Sportergebnisse, Nachrichten oder Börsenkurse).

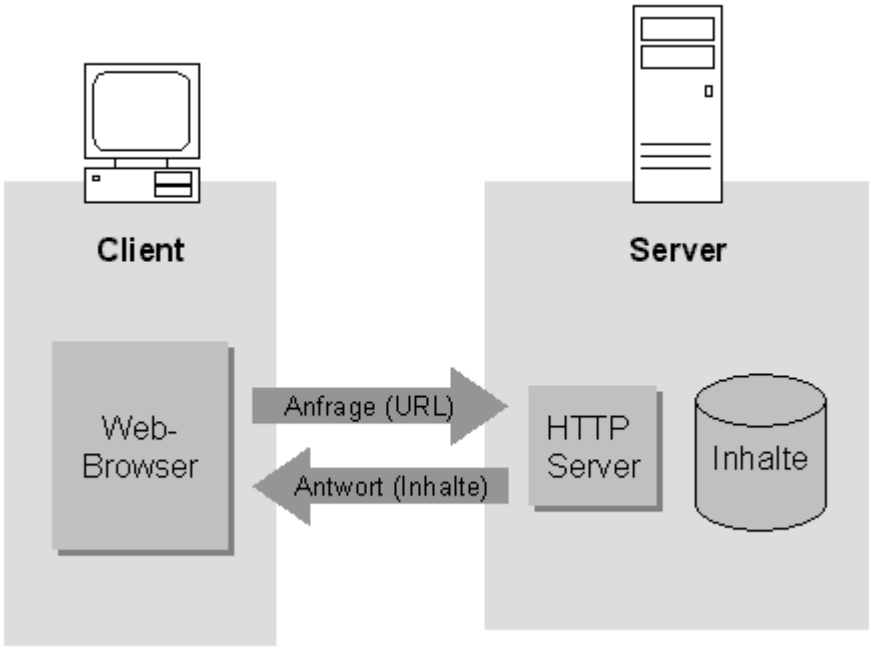

Abbildung 1: Das World-Wide-Web-Modell (modifiziert nach WAP Forum 2001a, Abbildung 1)

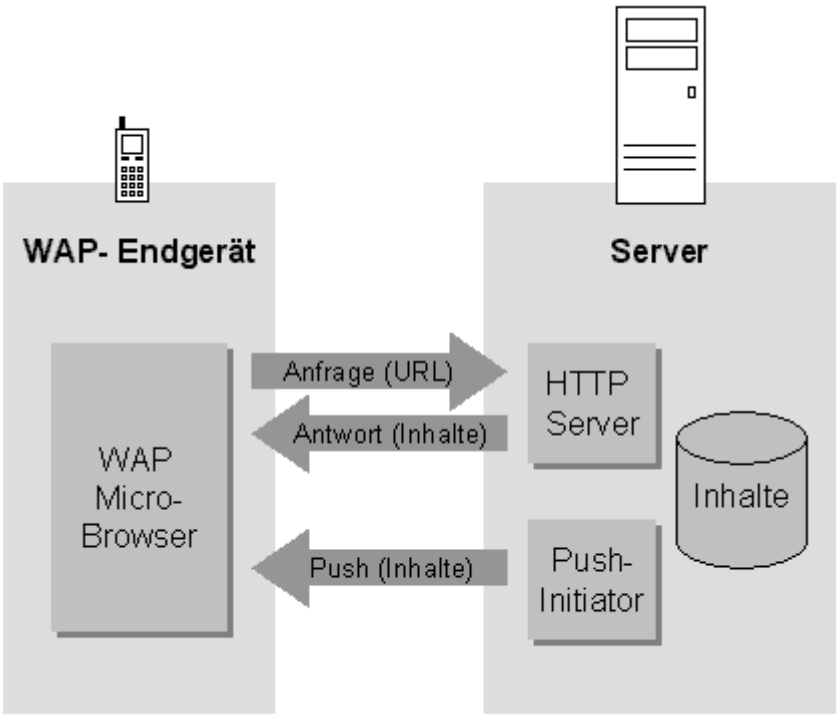

Abbildung 2: Das WAP-Modell (modifiziert nach WAP Forum 2001a, Abbildung 2)

Im Unterschied zu einem WWW-Client, hat ein WAP-Endgerät nie eine direkte Verbindung zum Server (s. Abbildung 3). Zwischen Endgerät und Server ist im Normalfall ein Gateway und meistens auch ein Proxy-Server geschaltet.

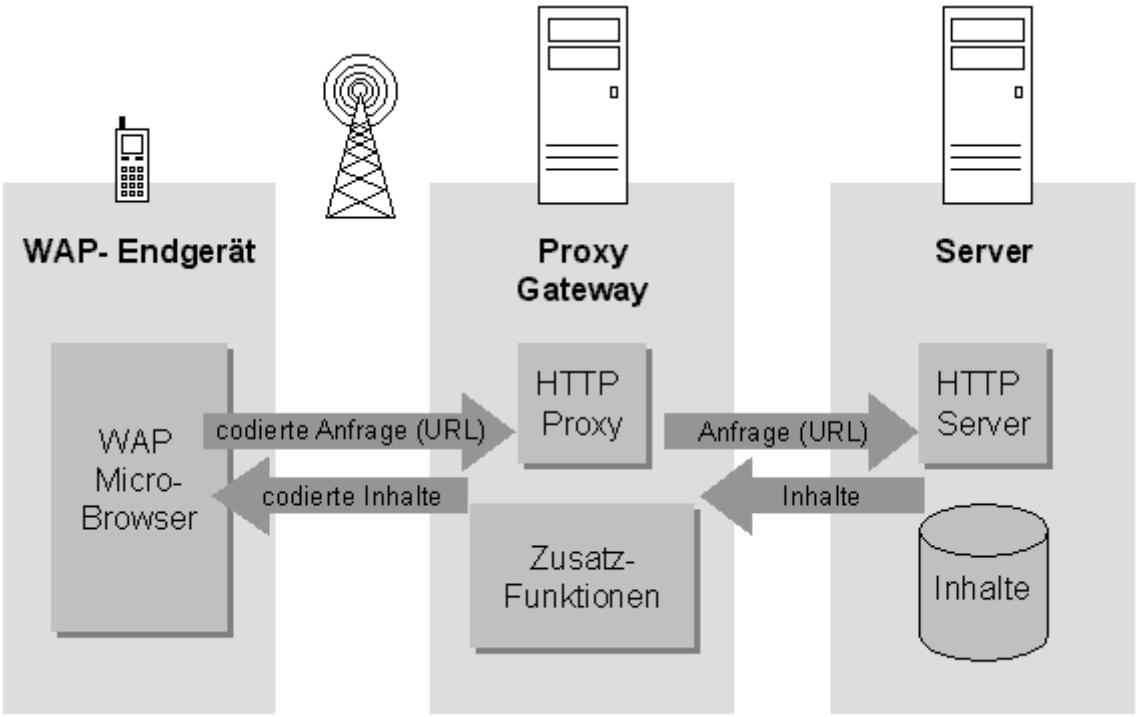

Abbildung 3: Anpassung des WWW-Modells durch Proxy- bzw. Gateway-Server (modifiziert nach WAP Forum 2001a, Abbildung 3)

Das Gateway hat zwei wichtige Aufgaben:

- Es setzt den WAP Protokoll-Stack in die Internet-Protokolle um (sofern WAP 1.x benutzt wird).
- Es codiert, decodiert bzw. kompiliert Daten auf ihrem Weg zwischen Endgerät und Server:
  - Anfragen werden vom Endgerät in binärer Form an das Gateway geschickt. Das Gateway decodiert die binäre Nachricht und schickt sie an den Server.
  - WML-Dokumente werden für die Übermittlung an das Endgerät in ein spezielles Binärformat konvertiert, das WAP Binary XML Content Format (WBXML) [4].
  - WMLScript-Dateien werden kompiliert. Sie werden dabei nicht wie WML-Dateien in ein kompakteres Format übersetzt, sondern in einen Bytecode umgewandelt. Auf dem Endgerät kann ein Virtual Machine (VM) Interpreter dann diesen Code ausführen. Dies ist vergleichbar mit der Art und Weise, wie Java-Code kompiliert wird, bevor er von der Java VM ausgeführt wird.

[4] In der Literatur und dem Nokia WAP Toolkit 2.0 wird die Umwandlung von WML-Dokumenten in Bytecode als Kompilierung bezeichnet. Da dieser Begriff normalerweise für die Übersetzung von einer Programmiersprache in eine andere benutzt wird, verzichte ich darauf, ihn im Zusammenhang mit einer Beschreibungssprache zu verwenden.

Durch die Codier-/Decodier-/Kompilierungs-Funktion des Gateways wird die Menge der Daten, die über das Mobilfunknetz übertragen werden muß, gering gehalten.

Wird ein Proxy-Server eingesetzt, dient er als Zwischenspeicher für häufig abgerufene Seiten. Anfragen des Endnutzers können so schneller bedient werden, da der Ursprungs-Server einer Ressource nicht mehr kontaktiert werden muß.

Der Applikations-Server und das Gateway können auch kombiniert werden. Hierbei spricht man dann von einem WAP-Server. Er kann beispielsweise in einem Unternehmensnetzwerk zum Einsatz kommen, wenn man sensible Daten nicht den öffentlichen Gateways der Netzprovider anvertrauen will.

4.2.3.2 Der WAP-Stack

Stack bedeutet eigentlich Stapel oder Stapelspeicher. In diesem Zusammenhang bezeichnet es die Gesamtheit der hierarchischen Schichten (und ihrer Protokolle) des WAP-Schichtenmodells.

Protokolle steuern in Netzwerken die Kommunikation zwischen dem Sender und dem Empfänger einer Nachricht. Wie Wenz/Hauser (2001) anmerken, besteht bei der Bezeichnung Protokoll etwas Verwirrung, da sie sowohl für ein einzelnes Protokoll (z. B. HTTP), aber auch für eine Protokollfamilie (z. B. WAP) verwendet wird.

Bei der Beschreibung der WAP-Protokollschichten muß man unterscheiden zwischen den WAP Versionen 1.x und 2.0. WAP 2.0 bietet mit dem Wireless Profiled HTTP (WP-HTTP) und dem Wireless Profiled TCP (WP-TCP) volle Interoperabilität mit dem herkömmlichen Internet-HTTP und TCP. Voraussetzung für den Einsatz dieser Protokolle ist allerdings, daß im Mobilfunknetzwerk die einzelnen Endgeräte über IP-Adressen angesprochen werden können.

Bei der Definition von WAP 1.x stand die Integration der WAP-Protokolle mit den Internet-Protokollen noch nicht im Vordergrund, da man vor allem versuchte, Protokolle zu entwickeln, die für Netzwerke mit geringer Bandbreite und hoher Latenzzeit geeignet waren.

Das WAP 1.x-Modell ist wie das OSI-Referenzmodell in hierarchischen Schichten aufgebaut. Jede Schicht kann mit der über und unter ihr liegenden Schicht kommunizieren. Anders als im OSI-Referenzmodell sind Anwendungs- und Darstellungsschicht zur einer Anwendungsschicht zusammengeschlossen und eine zusätzliche Sicherheitsschicht wurde eingeführt (siehe Abbildung 4).

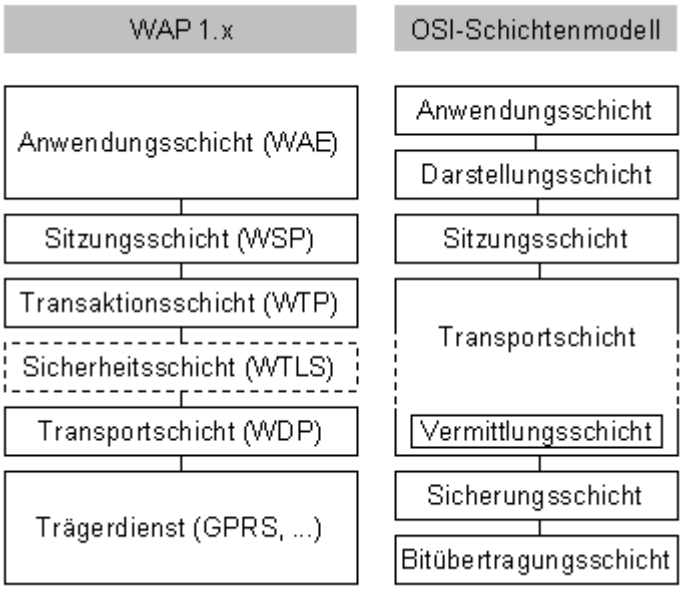

Abbildung 4: Das WAP 1.x-Schichtenmodell im Vergleich zum OSI-Schichtenmodell

Die einzelnen Schichten bzw. Protokolle des WAP 1.x-Modells sind:

- **Wireless Application Environment (WAE):** Aufgabe der WAE ist es, Inhalte darzustellen. Wichtige Elemente der WAE auf dem Endgerät sind daher der WML-Microbrowser (WAE User Agent) und der WTA User Agent. Zur WAE zählen außerdem die Sprachen WML und WMLScript, das Grafikformat WBMP, die WTA (WTA steht für Wireless Telephony Application, und erlaubt Zugriff auf Telefonfunktionen, mehr dazu in Kapitel 4.2.3.5). Auf Seiten des Web-Servers gehören noch die sogenannten Content Generators dazu, Anwendungen, die dynamische Inhalte erzeugen (PHP, ASP, ...).
- **Wireless Session Protocol (WSP):** Das WSP stellt verbindungsorientierte und verbindungslose Sitzungsdienste (session services) zur Verfügung. Im verbindungsorientierten Modus kann eine Sitzung pausiert und wiederaufgenommen werden, wenn nötig sogar von einem anderen Trägerdienst aus. Nachrichten, die den verbindungslosen Dienst nutzen, werden von der WSP direkt an die Schicht des Wireless Datagram Protocol (WDP) durchgereicht.
- **Wireless Transaction Protocol (WTP):** Zusammen mit dem darunterliegenden Wireless Datagram Protocol (WDP) bildet das WTP die Transportschicht im WAP-Schichtenmodell. Das WTP bietet 3 Arten von Transaktionsdiensten an:
  - unzuverlässige Ein-Weg-Anfragen (unreliable one-way request): Der Anfragende (Initiator) schickt eine Nachricht, der Empfänger (Responder) schickt keine Bestätigung. Einsatz findet diese Art der Transaktion vor allem bei Push-Diensten.

  - zuverlässige Ein-Weg-Anfragen (reliable one-way request): der Initiator schickt eine Nachricht, der Responder schickt eine Empfangsbestätigung und wartet kurz ab, ob der Initiator (evtl. fälschlicherweise) die Nachricht wiederholt.
  - zuverlässige Zwei-Weg-Anfragen mit Ergebnisnachricht (reliable two-way request with result message): Der Initiator schickt eine Nachricht, Der Responder bereitet das Ergebnis der Anfrage vor, die er gleich zurückschickt. Falls dies länger dauert kann er auch eine Bestätigung mit der Aufforderung, noch zu warten, versenden.

- **Wireless Transport Layer Security (WTLS):** Die WTLS ist nur eine Option. Sie basiert auf TSL v1.0, ein Standard, der wiederum auf SSL v3.0 zurückgeht. Die WTLS kommt nur dann zum Einsatz, wenn Vertraulichkeit, Integrität und Authentizität der gesendeten Daten gefordert wird.
- **Wireless Datagram Protocol (WDP):** Aufgabe des WDP ist, zu sendende Daten an den Trägerdienst zu adaptieren. Das WDP gleicht dabei die verschiedenen Fähigkeiten und Funktionen der Trägernetze aus (z. B. Bandbreite, Paketgrößen, IP-Unterstützung). Statt WDP kann auch UDP als Transportschicht eingesetzt werden.
- **Trägerdienste:** Das WAP Forum ist bestrebt, daß WAP alle gängigen Trägerdienste und -Netzwerke unterstützt. Hierzu zählen im GSM-Netz beispielsweise CSD (Circuit Switched Data), HSCSD (High-Speed Circuit Switched Data) und GPRS (General Packet Radio Service).

Mit WAP 2.0 bietet sich nun die Möglichkeit, in der Transportschicht TCP und der Sitzungsschicht HTTP/1.1 einzusetzen. Voraussetzung für den Einsatz von TCP ist, daß der Trägerdienst Daten über das Internet Protocol (IP) transportieren kann, wie es bei GPRS und UMTS vorgesehen ist. Die Wireless-Profiled-Varianten des HTTP und des TCP sind voll kompatibel mit den im Internet bzw. World Wide Web verwendeten Varianten. Abbildung 5 verdeutlicht die durch WAP Version 2.0 erreichte Analogie von WAP- und WWW-Modell.

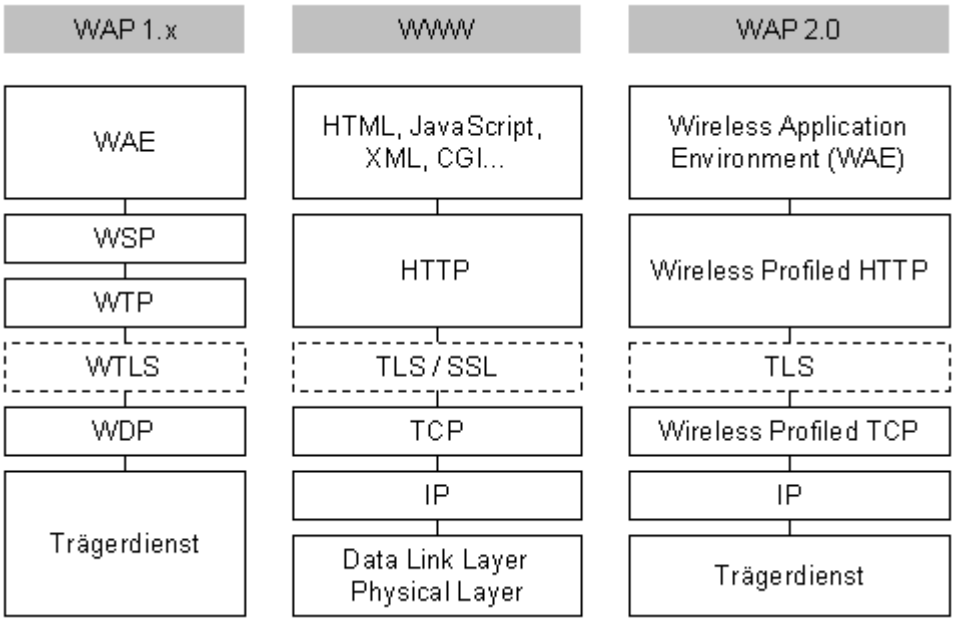

Abbildung 5: Die WAP-Modelle und das WWW-Modell

Wird der WAP 2.0-Stack vollständig angewendet, kann ein WAP-Gateway überflüssig werden, da die Umsetzung von Protokollen nicht mehr notwendig ist.

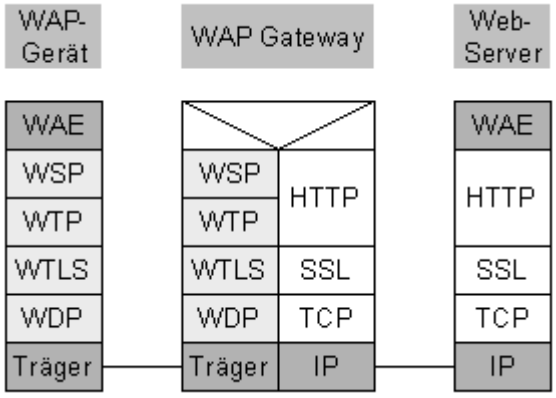

Abbildung 6: WAP 1.x-Gateway (modifiziert nach WAP Forum 2001a, Abbildung 8)

Abbildung 6 zeigt das noch in WAP-1.x notwendige typische WAP-Gateway. Bei der Konvertierung der WAP-Protokolle in die üblichen Internet/WWW-Protokolle ist zu beachten, daß WAP auf der Transportschicht verbindungslos auf der Basis von Datengrammen (Wireless Datagram Protocol), das Internet dagegen verbindungsorientert arbeitet (Transmission Control Protocol).

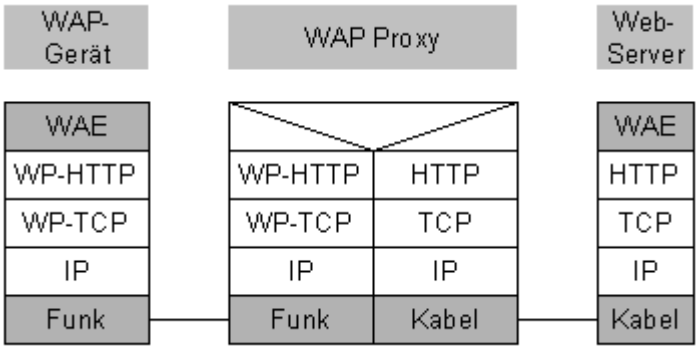

Abbildung 7: WAP-HTTP-Proxy mit Wireless Profiled TCP und HTTP (modifiziert nach WAP Forum 2001a, Abbildung 9)

WAP 2.0 ermöglicht dien in Abbildung 7 gezeigte Beispielkonfiguration. Hier wird ein WAP-Proxy eingesetzt. Wie bei einem normalerweise im WWW benutzten Proxy dient er zur temporären Speicherung (Caching) von Webseiten, die ein Nutzer angefordert hat, um sie anderen Nutzern schneller zur Verfügung stellen zu können. In dieser Konfiguration ist der WAP Proxy zwischen dem funk- und dem kabelgebundenen Netzwerk platziert. Er bietet dadurch den zusätzlichen Nutzen, daß er im Funknetzwerk die Vorteile des Wireless Profiled HTTP und TCP ausnutzen kann.

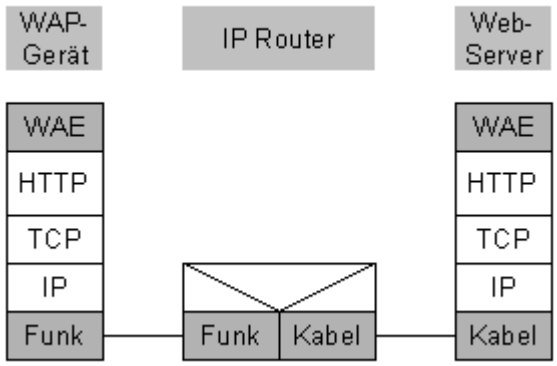

Abbildung 8: Direkter Zugriff von einem WAP Gerät auf den Web-Server (modifiziert nach WAP Forum 2001a, Abbildung 11)

Da WP-HTTP und WP-TCP sich nur durch einige auf das Funknetz abgestimmte Zusätze von HTTP und TCP unterscheiden, ist auch ein direkter Zugriff von einem WAP-Gerät auf einen Web-Server denkbar. Es ist lediglich ein IP-Router nötig, der die Verbindung zwischen Funk- und Kabel-Netzwerk darstellt (Abbildung 8). Allerdings muß man bei dieser Konfiguration auf die Vorteile der auf das Funknetz optimierten Protokolle verzichten.

Da die Umstellung von WAP 1.x auf WAP 2.0 nach und nach vor sich gehen wird, sieht das WAP Forum Endgeräte vor, die sowohl den WAP 1.x als auch den WAP 2.0-Stack unterstützen (Dual Stack Support).

4.2.3.3 Wireless Markup Language (WML)

Um Dokumente in einem WAP-Browser darzustellen, wurde die Beschreibungssprache WML (Wireless Markup Language) entwickelt. Sie ist das WAP-Pendant zu HTML. Wie in HTML gibt es die Möglichkeit, Text, Tabellen, Formulare und Bilder darzustellen, sowie über Hyperlinks zwischen Dokumenten und Dokumentteilen zu navigieren. Die zweite Version der WML (im Folgenden WML2 genannt, im Gegensatz zu WML1) wird definiert in der Spezifikation WAP-238-WML-20010626-p (WAP Forum 2001b).

WML wurde mit WAP Version 1.0 eingeführt, in WAP 1.1 ergänzt, und in WAP 2.0 wesentlich verändert. WML ist eine Untergruppe von XML, der Extended Markup Language, dem W3C-Standard für alle Beschreibungssprachen des Internet. Während die ersten Versionen von WML eine recht eigenständige Sprache bildeten, die besonders auf die Bedürfnisse des Mobilfunks abgestimmt war, schließt die WML2-Spezifikation XHTML Basic (Extended Hypertext Markup Language) ein, die Neudefinition von HTML in XML, sowie eine Untergruppe (Subset) von CSS (Cascading Stylesheets), eine Formatvorlagenbeschreibungssprache, um die Darstellung von WML2-Seiten zu vereinheitlichen.

Zur Zeit gibt es noch keine Endgeräte, die WML2 unterstützen, daher wird im Folgenden nur WML1 berücksichtigt. WML2 ist vollständig rückwärts-kompatibel zu WML1, auch wenn das WAP Forum dazu rät, den WML2-Sprachelementen den Vorzug zu geben.

WML ist, wie schon erwähnt, eine Untersprache von XML. Daher hat es die gleiche Tag-Syntax. Tags sind bestimmte Zeichenketten, die den Browser veranlassen, die darin eingeschlossenen Dokumentteile in einer bestimmten Art darzustellen, bzw. Zeilenumbrüche oder Bilder einzufügen. In WML werden Tags wie folgt abgeschlossen:

`<tag> ... </endtag>`	Start und Endtag
`<emptytag/>`	in sich abgeschlossener Leertag

Auf den ersten Blick gibt es viele Ähnlichkeiten zwischen WML und HTML, bei näherer Betrachtung stellen sich jedoch einige entscheidende Unterschiede heraus. Seeboerger-Weichselbaum (2000, S. 37-40) bietet eine recht umfassende Übersicht über die Unterschiede und Gemeinsamkeiten von WML und HTML sowie WML-spezifische Erweiterungen. Als wichtigste Unterschiede seien genannt:

- WML ist ungleich strikter als HTML. HTML-Browser sind sehr fehlertolerant. Es macht im Normalfall keinen Unterschied, ob `<img width=“100“>` oder `<img width=100>` geschrieben wird, der Browser interpretiert die Angabe richtig.
- WML verlangt nach Tags in Kleinbuchstaben und Anführungszeichen um Attributsangaben.

- Zu jedem Tag gibt es einen Ende-Tag. Leertags müssen in sich abgeschlossen werden. Ausnahmen sind die beiden Tags `<?xml>` und `<!DOCTYPE>`.
- Jede WML-Datei beginnt mit dem Prolog mit den Tags `<?xml>` und `<!DOCTYPE>`. Im `<!DOCTYPE>` Tag wird in der Document Type Definition (DTD) die verwendete WML-Version angegeben.
- WML-Dokumente haben eine andere Struktur als HTML-Dokumente. Sie bestehen nicht aus einer Seite sondern aus mehreren Karten (Cards), aber dazu später mehr.
- In WML gibt es die Möglichkeit, über das `<access>`-Tag den Zugriff auf das Dokument zu begrenzen. Es kann dann nur noch über Hyperlinks in WML-Dokumenten angesprungen werden, die in einer bestimmten Domain oder gar in einem bestimmten Pfad dieser Domain liegen.
- In WML können Variablen gesetzt werden. So können Benutzereingaben in einem Formular im Text des WML-Dokumentes angezeigt werden.

Der typische Aufbau eines WML-Dokumentes ist vergleichbar mit einem Stapel (Spiel-)Karten (siehe Abbildung 9). Ein WML-Dokument wird daher oft auch als WML-Deck bezeichnet (von engl. Deck = Kartenspiel, Kartenstapel).

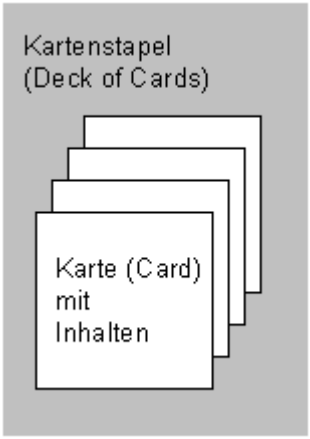

Abbildung 9: WML-Dokument als Kartenstapel (WML-Deck)

Über Hyperlinks können die einzelnen Karten eines Stapels angesprungen werden. Der Grund für diese Struktur liegt in der langsamen Übertragungsgeschwindigkeit der Mobilfunknetzwerke und der recht langen Zeit, die benötigt wird, um Kontakt zum Gateway aufzunehmen. Der WML-Entwickler kann dem Benutzer WML-Seitenabrufe ersparen, indem er ihm in einem WML-Deck nicht nur die unmittelbar gewünschte Information übermittelt, sondern zusätzlich Informationen, die der Benutzer mit hoher Wahrscheinlichkeit ebenfalls erhalten will. Diese sind für den Benutzer vorerst unsichtbar, erst wenn er die entsprechenden Hyperlinks aktiviert, wird die entsprechende Karte angezeigt.

Am einfachsten lässt es sich anhand eines Beispiels darstellen: Nehmen wir einmal an, ein Benutzer will Nachrichten abrufen. Im World Wide Web würde ihm zuerst ein HTML-Dokument präsentiert, das Schlagzeilen (als Hyperlinks) enthält. Mit der Auswahl eines Hyperlinks ruft er ein weiteres HTML-Dokument ab, das den gewünschten Artikel zur Schlagzeile enthält. Nun will er vielleicht eine weitere Nachricht lesen. Er kehrt zurück zu seiner Ausgangsseite und ruft von dort das nächste HTML-Dokument auf. In einem WML-Deck könnte ihm auf der ersten Karte die Liste der Schlagzeilen angezeigt werden, während die weiteren Karten bereits die Nachrichtenartikel enthalten. Wenn er über die Schlagzeilen eine der Nachrichten anspringt, wird sie ohne Verzögerung angezeigt, da sie bereits im Speicher des Endgerätes liegt und nicht mehr vom Server abgerufen werden muß.

Ein WML-Entwickler, der diese Möglichkeit nutzt, sollte versuchen, einen guten Kompromiss zu finden zwischen der Datenmenge, die an den Benutzer übermittelt wird, und dem Anteil dieser Daten, die der Benutzer überhaupt betrachten wird. Der Komfortgewinn, den der Nutzer durch das Vorhersehen seiner Bewegungen in einem WML-Angebot hat, sollte nicht dadurch vernichtet werden, daß ihm bei der Übermittlung des ersten WML-Decks eine zu lange Ladezeit zugemutet wird.

Wie sieht diese Seitenstruktur nun im Quelltext aus? Hier folgt das typische Gerüst einer WML-Seite, dessen Elemente nachher im einzelnen erläutert werden:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http.//www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <head>
    <!-- Kopfbereich -->
  </head>
  <template>
    <!-- Vorlagen für Cards -->
  </template>
  <card>
    <!-- Card -->
  </card>
  <card>
    <!-- weitere Card, beliebig viele möglich -->
  </card>
...
</wml>
```

- `<?xml version="1.0"?>`: Jede WML-Datei muß mit diesem Tag beginnen. Damit wird sie als XML-Dokument deklariert.
- `<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http.//www.wapforum.org/DTD/wml_1.1.xml">`: Hier wird auf die Document Type Definition (DTD) für die WML-Seite verwiesen. In der DTD ist die Form ei-

ner WML-Seite beschrieben, welche Tags es gibt, welche Eigenschaften sie annehmen können und wo sie erlaubt sind. In diesem Fall handelt es sich um eine Seite in WML Version 1.1. Für WML Version 1.2, Version 1.3 oder Version 2 heißt es analog:

```
<!DOCTYPE wml PUBLIC “-//WAPFORUM//DTD WML 1.2//EN“
“http.//www.wapforum.org/DTD/wml11.xml“>
<!DOCTYPE wml PUBLIC “-//WAPFORUM//DTD WML 1.3//EN“
“http.//www.wapforum.org/DTD/wml13.xml“>
<!DOCTYPE html PUBLIC"-//WAPFORUM//DTD WML 2.0//EN"
"http://www.wapforum.org/wml20.dtd">
```

- `<wml> ... </wml>`: Vergleichbar mit dem `<html>...</html>`-Tag. Es umschließt das eigentliche Dokument.
- `<head> ... </head>`: Dies ist ein optionaler Abschnitt. Er kann einen Seitentitel, Metaangaben (z. B. für Suchmaschinen) und /oder Zugriffsbeschränkungen enthalten.
- `<template> ... </template>`: Ebenfalls ein optionaler Teil. Hier können Vorlagen definiert werden, die für alle Karten des Deck gelten, wie z. B. „Zurück“-Links, die von jeder Card wieder zurück zur Anfangscard führen.
- `<card> ... </card>`: Vergleichbar mit dem `<body>`-Tag in HTML. Zwischen diesen Tags steht der eigentliche Inhalt eines WML-Dokumentes.

Tabelle 1: Wichtige WML-Tags

WML-Tag	**vgl. HTML**	**Bedeutung**
`<?xml version=“1.0“?>`	-	Kennzeichnung als XML-Dokument
`<!DOCTYPE ...>`	`<!DOCTYPE ...>`	Verweis auf die DTD
`<!--...-->`	`<!--...-->`	Kommentar

`<head></head>`
`<title></title>`
`<meta/>`
`<access/>`
	`<head></head>`
	`<title></title>`
	`<meta>`
	-
		Seitenkopf
		Seitentitel
		Metaangabe
		Zugriffsbeschränkung

`<wml></wml>`	`<html>...</html>`	Umschließt das eigentliche Dokument
`<template>` `</template>`	-	Vorlage für alle Cards (z. B. Links)
`<card></card>`	`<body></body>`	Umschließt den eigentlichen Inhalt
`<p></p>` ` `	`<p></p>` ` `	Absatz im Text Zeilenumbruch
`<img src=“”/>`	`<img src=“”>`	Einfügen eines Bildes

<a href=“”/>	<a href=“”>	Hyperlink
<table></table> <tr></tr> <td></td>	<table></table> <tr></tr> <td></td>	Tabelle Tabellenzeile Tabellenzelle
  <u></u> <big></big>	 <u></u> <big></big>	Textformatierung (fett, kursiv,unterstrichen, klein, groß)
<input.../>	<input...>	Eingabefeld im Formular (in WML gibt es keinen <form>-Tag)
<select> <option></option> </select>	<select> <option> </select>	Auswahlliste im Formular
<do></do>	<input type=”button”>	Eine Taste anzeigen lassen. Bei einigen Endgeräten wird statt einer Taste allerdings ein Untermenü angezeigt.
<timer/>	-	Timer, der nach einer bestimmten Zeit eine Aktion ausführt (z. B. nächste Card zeigen)
<onevent> </onevent>	-	Eventhandler
<setvar/>	-	Variablen setzen

Tabelle 1 listet alle wichtigen WML-Tags und die eventuell vergleichbaren HTML-Tags auf. Für weitere Informationen zur WML-Syntax möchte ich auf die einschlägige Literatur hinweisen, insbesondere Wenz/Hauser (2001) und Seeboerger-Weichselbaum (2000) sind empfehlenswert.

WML-Dokumte werden im Textformat mit der Endung „.wml“ abgespeichert. Sind sie bereits in das Binärformat umgewandelt, bekommen sie die Endung „.wmlc“. Dies ist jedoch nur in Sonderfällen nötig, da zur Zeit ein Endgerät normalerweise über ein WAP-Gateway die Verbindung zu einem Server aufnimmt, wobei das Gateway die Kovertierung in WBXML übernimmt. (s. a. Kapitel 4.2.3.1)

Wie ein WML-Dokument in WBXML (WAP Binary XML Content Format) umgewandelt wird, definiert die Spezifikation WAP-192-WBXML-2001-0725-a (WAP Forum 2001e): WML-Tags werden durch einzelne Bytes ersetzt, versehen mit den Zusatzinformationen, ob das Tag Attribute hat, ob ein Ende-Tag nötig ist und ob das Tag einen Inhalt hat. Kommentare, der für User Agents uninteressante Teil der Metaangaben sowie überflüssige Leerzeichen werden entfernt.

Das folgende WML-Beispiel-Deck hat als WML-Datei abgespeichert eine Größe von 948 Bytes.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>

  <template>
    <!-- Template: auf jeder Card wird ein "Zurück"-Link erscheinen.
-->
    <do type="prev"><prev/></do>
  </template>

  <card id="card1" title="Erste Card">
    <p align="center">
      <!-- Inhalt der ersten Card -->
      <big><b>Dies ist die Erste Card.</b></big>
</p>
  <p align="left">
     <!-- Link zur zweiten Card -->
    <small><a href="#card2">Hier geht es zur zweiten Card.</a></small>
  </p>
  </card>

  <card id="card2" title="Zweite Card">
  <!-- Timer-Event: nach 2 Sekunden wird wieder die Erste Card
angezeigt -->
  <onevent type="ontimer">
      <prev/>
    </onevent>
    <timer value="20"/>
  <!-- Inhalt der zweiten Card. -->
    <p align="center">
      <big><b>Dies ist die Zweite Card.</b></big>
    </p>
  </card>
</wml>
```

Nach der Konvertierung ergibt sich eine Länge von nur 193 Bytes. (Die Konvertierung wurde in diesem Fall mit der Entwicklungsumgebung Nokia Mobile Internet Toolkit, Version 3.0 durchgeführt.)

```
02 09 6A 00 7F 7B E8 46 01 32 01 01 E7 55 03 63  ..j..{.F.2...U.c
61 72 64 31 00 36 03 45 72 73 74 65 20 43 61 72  ard1.6.Erste Car
64 00 01 E0 07 01 65 64 03 44 69 65 73 20 69 73  d.....ed.Dies is
74 20 64 69 65 20 45 72 73 74 65 20 43 61 72 64  t die Erste Card
2E 00 01 01 01 E0 08 01 78 DC 4A 03 23 63 61 72  ........x.J.#car
64 32 00 01 03 48 69 65 72 20 67 65 68 74 20 65  d2...Hier geht e
73 20 7A 75 72 20 7A 77 65 69 74 65 6E 20 43 61  s zur zweiten Ca
72 64 2E 00 01 01 01 01 E7 55 03 63 61 72 64 32  rd.......U.card2
00 36 03 5A 77 65 69 74 65 20 43 61 72 64 00 01  .6.Zweite Card..
```

```
F3 3F 01 32 01 BC 4D 03 32 30 00 01 E0 07 01 65  .?.2..M.20.....e
64 03 44 69 65 73 20 69 73 74 20 64 69 65 20 5A  d.Dies ist die Z
77 65 69 74 65 20 43 61 72 64 2E 00 01 01 01 01  weite Card......
01                                               .
```

4.2.3.4 WMLScript

Das Wireless Application Protocol erlaubt es dem Entwickler, WML-Dokumente dynamischer und interaktiver zu gestalten, indem er in der Skriptsprache WMLScript verfasste Funktionen einbindet. WMLScript wird auf dem Client ausgeführt. Dort steht nur wenig Speicherplatz und Rechenleistung zur Verfügung, daher ist WMLScript in seinem Funktionsumfang im Vergleich beispielsweise zu JavaScript recht eingeschränkt.

Die Haupteinsatzgebiete von WMLScript sind:

- Überprüfung von Benutzereingaben in Formularfeldern
- Zugriff auf Telefonfunktionen, wie z. B. Tätigen eines Anrufes oder Ablegen von Einträgen im internen Telefonbuch.
- Erzeugen und Ausgeben von Meldungen und Dialogen lokal auf dem Endgerät

WMLScript ist eine erweiterte Untergruppe (Subset) von JavaScript. Wie JavaScript basiert es auf ECMAScript. ECMAScript (auch ECMA-262) wurde von der European Computer Manufacturer Association entwickelt, als Standard für Internet-Skriptsprachen. Heijden/Frost (2000, Tabelle 2.1) und Seeboerger-Weichselbaum (2000, S. 136/137) bieten eine umfangreiche Übersicht über Gemeinsamkeiten und Unterschiede von WMLScript und ECMAScript bzw. JavaScript. Als wichtigste Punkte seien genannt:

- WMLScript ist nicht objektorientiert bzw. objektorientiert-basiert. Seine Bibliotheken (z. B. Dialogs) dürfen nicht mit Objekten verwechselt werden.
- WMLScript unterscheidet wie JavaScript bei der Variablendeklaration nicht zwischen verschiedenen Datentypen.
- WMLScript unterstützt keine Arrays.
- In WML-Script fehlt die Schleife `do-while` und die Abfrage `switch-case`.
- Für Umsteiger interessant: die generelle Syntax, die Operatoren und viele Ausdrücke (Statements) in WMLScript ist bzw. sind identisch mit JavaScript.

WMLScript-Funktionen werden, im Gegensatz zu JavaScript-Funktionen, immer in einer eigenen Datei mit der Endung „.wmls“ abgelegt. Sie lassen sich nicht im WML-Code einbinden. In einer WMLScript-Datei können mehrere Funktionen gespeichert werden. Der Aufruf einer Funktion aus einem WML-Dokument erfolgt über einen Link: `<a href=“datei.wmls#funktion(parameter)“>` . Damit eine Funktion so aufgerufen werden kann, muß sie als `extern function` gekennzeichnet sein, ansonsten kann sie nur von Funktionen innerhalb der WMLScript-Datei aufgerufen werden.

WMLScript verfügt über sieben Standard-Bibliotheken (Libraries). Sie sind definiert in WAP-194-WMLScriptLibraries-20000925-a (WAP Forum, 2000b). Der Aufruf einer Funktion aus einer Bibliothek erfolgt über `Bibliotheksname.funktion()`, z. B. `Dialogs.alert()`.

- `Lang`: enthält Kernfunktionen, wie Funktionsabbruch (`Lang.abort(errorDescription)`) oder das Durchsuchen einer Zeichenkette nach einer Zahl (`Lang.parseInt(value) bzw. Lang.parseFloat(value)`).
- `Float`: enthält mathematische Fließkomma-Operationen, z. B. Runden einer Zahl (`Float.round()`). Die Float-Bibliothek ist optional, da einige Endgeräte keine Fließkomma-Operationen unterstützen.
- `String`: enthält Funktionen für die Arbeit mit Strings (Zeichenketten), z. B. Rückgabe der Länge (`String.length(string)`) oder Vergleich von Strings (`String.compare(string1, string2)`).
- `URL`: enthält Funktionen für die Arbeit mit absoluten und relativen URLs, z. B. Überprüfung der URL-Syntax auf Gültigkeit (`URL.isValid(url)`), Extraktion von Protokoll, Host, Port, Pfad, Parametern, Ankern.
- `WMLBrowser`: enthält Funktionen, die einer WMLScript-Funktion erlauben, auf den WML-Kontext, aus dem sie aufgerufen wurde, zuzugreifen. Beispielsweise erlauben sie das Auslesen bzw. Setzen von WML-Variablen (`WMLBrowser.getVar(name)` bzw. `WMLBrowser.setVar(value, name)`) oder den Aufruf einer URL (`WMLBrowser.go(url)`).
- `Dialogs`: enthält eine Anzahl typischer Funktionen, die für eine Benutzeroberfläche nützlich sind, z. B. Eingabedialoge (`Dialogs.prompt(message, defaultInput)`), Bestätigungsmeldungen (`Dialogs.confirm(message, ok, cancel)`) und Warnmeldungen (`Dialogs.alert(message)`).
- `Crypto`: Diese Bibliothek wurde erst im Juni 2001 hinzugefügt. Sie ist in der eigenen Spezifikation WAP-161-WMLScriptCrypto-20010620-a (WAP Forum 2001f) definiert und enthält bisher Funktionen für die digitale Signierung. Funktionen zur Ver- und Entschlüsselung können noch folgen.

Allgemein ist zu WMLScript noch anzumerken, daß es nicht von allen Endgeräten unterstützt wird. Der UP.Browser von Openwave, der in vielen Mobiltelefonen verwendet wird, unterstützt WMLScript erst ab Version 4. Auf dem europäischen Markt sind davon das Modell One Touch von Alcatel, sowie das Samsung SGH-800 betroffen. Die meisten WML-Browser für den PC (z. B. WinWAP 3.0 PRO) und Mobiltelefon-Simulatoren unterstützen WMLScript ebenfalls nicht (z. B. Wireless Companion Release 2.51, der Online-Simulator von http://www.gelon.net. Je nachdem, welche Endgeräte die Zielgruppe der WAP-Anwendung benutzt, sollte man überprüfen, wie sich die Nicht-Unterstützung von WMLScript auf die Kernfunktionalität der Anwendung auswirkt.

Abschließend noch eine Anmerkung zur Sicherheit des Einsatzes von WMLScript. Im Gegensatz zu Internet-Skript- und Programmiersprachen (JavaScript, Active Scripting/Active X, Java-Applets), sind in WMLScript bisher keine Sicherheitslücken bekannt geworden.

4.2.3.5 Wireless Telephony Application (WTA)

Es gibt verschiedene Möglichkeiten, WML-Dokumente zu betrachten, beispielsweise über einen PC-WML-Browser, mit einem PDA, aber das am häufigsten benutzte Gerät ist immer noch das Mobiltelefon. Das WAP Forum hatte bei der Entwicklung von WAP zwar das Ziel, einen mobilen Zugriff auf Internet-Informationen zu ermöglichen, es vergaß jedoch nicht, die Chancen, die sich aus der Integration von Internet und Telefonie ergeben, zu nutzen. Hier ist der Ansatzpunkt der Wireless Telephony Application (WTA). Sie erweitert den WAE User Agent (WML-Browser) um den WTA User Agent, der Zugriff auf die Funktionen des Telefons bzw. des Mobilfunknetzes erlaubt. Die Schnittstelle, über die dieser Zugriff erfolgt, ist das Wireless Telephony Application Interface (WTAI).

Typische Funktionen, die vom WTAI zur Verfügung gestellt werden, sind die Anruf-Behandlung (Tätigen, Entgegennehmen), das Verwalten des Telefonbuchs und das Anzeigen und Senden von SMS-Texten.

Ein mögliches Szenario für den Einsatz der WTAI-Funktionen wäre eine Callcenter- bzw. Telefonbuchanwendung: Der Benutzer sucht aus einem Verzeichnis (in WML) den passenden Ansprechpartner und bekommt sofort angeboten, eine Verbindung zu der angezeigten Telefonnummer herzustellen.

Die WTAI-Funktionen sind in mehreren Bibliotheken verfügbar, die sich wiederum drei Kategorien zuordnen lassen:

- **Public WTAI:** einfache Funktionen, die über den WAE User Agent auch Applikationen Dritter zur Verfügung gestellt werden. Alle diese Funktionen sind in der Bibliothek Public WTAI (WTAPublic) gespeichert: Anrufen, DTMF (Wähltöne) senden und Einträge zum Telefonbuch hinzufügen.
- **Network Common WTAI**: Funktionen, die normalerweise jedes Mobilfunk- bzw. Telefonnetz unterstützt. Sie sind nur über den WTA User Agent zugänglich. Sie gliedern sich in die Bibliotheken Network Common WTAI Voice Call Control (WTAVoiceCall) zur Anrufbehandlung und zum DTMF-Senden, Network Common WTAI Network Messages/Text (WTANetText) zum Senden und Anzeigen von SMS-Texten, Network Common WTAI Phonebook (WTAPhoneBook) für den Zugriff auf das Telefonbuch, Network Common WTAI Call Logs (WTACallLog) für Zugriff auf Anruflisten und Network Common WTAI Miscellaneous (WTA-Misc) für diverse andere Funktionen (z. B. Netzwerkstatus)
- **Network Specific WTAI**: Funktionen, die nur in spezielle Netzwerken funktionieren oder vom Netzwerk-Operator speziell zur Verfügung gestellt wurden.

In den Spezifikationen für WTA (WAP Forum 2001c) bzw. WTAI (WAP Forum 2001d) ist das Sicherheitskonzept beschrieben, das gewährleisten soll, daß auf dem Mobiltelefon nur vom Mobilfunknetz-Betreiber freigegebene WTA-Dienste in Anspruch genommen werden können. Kernpunkt des Sicherheitsmodells ist, daß es vertrauenswürdige Gateways gibt (z. B. die offiziellen Gateways der Mobilfunk-Provider) und daß Gateway-Besitzer entscheiden, welchen Servern sie erlauben, auf WTAI-Funktionen zuzugreifen.

Der Aufruf von WTAI-Funktionen kann in WML über einen Uniform Resource Identifier (URI) erfolgen, bei einigen Funktionen ist es auch über eine WMLScript-Funktion möglich. Beispielsweise kann der Aufbau einer Telefon-Verbindung aus einer WML-Seite über folgenden Link geschehen:

```
<a href="wtai://wp/mc;0123456789">Waehlen</a>
```

Jeder URI-Aufruf einer Funktion beginnt mit der Angabe des ‚Protokolls' wtai://, dann folgt das Kürzel der WTAI-Bibliothek, das Kürzel der Funktion, sowie die Parameter. Die WMLScript-Variante sieht wie folgt aus:

```
var phone = WTAPublic.makeCall(„0123456789");
```

Zuerst steht der Name der Bibliothek, dann der der WTAI-Funktion, der die Telefonnummer als Parameter übergeben wird.

Einige der WTAI-Funktionen haben einen Rückgabewert, der über eventuell aufgetretene Fehler informiert, auf die die WAP-Applikation reagieren kann. Liefert das obige Beispiel „-105" zurück, ist die Gegenstelle besetzt. Dem Benutzer könnte nun angeboten werden, die Nummer für einen weiteren Anrufversuch abzuspeichern.

Für weitere Informationen zum WTAI und seinen Funktionen möchte ich wieder auf die Literatur verweisen, insbesondere Seeboerger-Weichselbaum (2000).

4.2.3.6 Wireless Bitmap (WBMP)

WAP-Angebote müssen nicht nur aus textuellen Inhalten bestehen. In den WAP-Spezifikationen ist auch ein Grafikformat vorgesehen (WAP Forum 2001g).

Wireless Bitmap (WBMP) ist ein recht einfaches Format. Es hat die Farbtiefe 1 Bit (schwarz-weiß) und kennt keine Kompression. Eine WBMP-Datei besteht aus zwei Teilen: dem Header (Kopfteil) und dem Beschreibungsteil. Der Header enthält Informationen über den WBMP-Typ (bisher gibt es nur Level 0),die Höhe und Breite des Bildes. Im Beschreibungsteil steht ein Bit jeweils für ein Pixel des Bildes (0 entspricht schwarz, 1 entspricht weiß). Auf das Hauptbild können noch bis zu 15 Bilder folgen, die zusammen eine Animation bilden.

Im Gegensatz zu WML und WMLScript wurde bei WBMP auf komprimierende Maßnahmen verzichtet. Wenz/Hauser (2000) vermuten, daß dies aus Rücksicht auf die geringe Rechenkapazität der Endgeräte geschieht.

4.3 WAP-Endgeräte

WAP wurde für Mobiltelefone entwickelt, daher machen sie den größten Anteil an den Endgeräten aus. Allgemein lassen sich WAP-Endgeräte in zwei Kategorien teilen:

- Geräte mit integriertem WML-Browser, meist ein Microbrowser, d. h. ein sehr schlank programmierter Browser.
- Geräten, die erst durch die Installation eines WML-Browsers zu WAP-Endgeräten werden.

4.3.1 Mobiltelefone

Mobiltelefone gehören zur ersten Kategorie. Sie waren die ersten WAP-Endgeräte (genauer gesagt, das Nokia 7110), schließlich wurde das Wireless Application Protocol speziell für sie entwickelt. Jeder Mobiltelefonhersteller hat inzwischen ein WAP-fähiges Modell in seiner Produktpalette. Nicht immer ist dies eine vollständige Eigenentwicklung: Siemens, Alcatel, Motorola, Panasonic und Samsung haben den UP.Browser, der von Phone.com/Openwave Systems entwickelt wird, lizenziert und in einige oder alle ihre Geräte integriert. Nokia, Trium und Ericsson (nicht alle Modelle) beispielsweise setzen dagegen auf Eigenentwicklungen.

Dies hilft natürlich nicht besonders dabei, WML-Dokumente zu erstellen, die auf allen Telefonen gleich dargestellt werden. Die Spezifikationen des WAP Forums mögen eng gefasst sein, aber nicht einmal die Modelle mit integriertem UP.Browser haben eine einheitliche Darstellung. Einerseits liegt das an den verschiedenen eingesetzten Versionen des UP.Browsers, andererseits auch an unvollständigen Implementationen der Hersteller.

Das Siemens S35 und das Motorola V.2288 enthalten beide die gleiche Browser-Version, jedoch erkennt das Motorola-Modell keinen über `<big>` formatierten Text, den das Siemens-Modell tadellos darstellt.

Auch fordert das WAP Forum, daß die Telefone zwei frei belegbare Tasten anbieten, denen über das Display eine Funktion zugewiesen werden kann, um beispielsweise einen „Zurück“-Link immer eingeblendet zu haben (sog. Softkeys). Beim WAP-Erstling Nokia 7110 findet man Einträge, die für diese Tasten vorgesehen waren, versteckt in einem Menü.

Auch scheinen die Hersteller bestrebt zu sein, eigene WML-Erweiterungen und Zusatzfunktionen zu entwickeln, die vom WAP Forum (bisher) nicht vorgesehen waren.

Der UP.Browser von Phone.com/Openwave Systems bietet eine Anzahl proprietärer Tags, z. B. `<exit>` (zum Beenden einer Card und Löschen der von ihr gesetzten Variablen) oder `<link>` (zum Einbinden eines WML-Dokumentes in ein anderes, ähnlich dem `<link>`-Tag in HTML). Seeboerger-Weichselbaum (2000) weist in der WML-Referenz seines Buches die UP.Browser-Erweiterungen nach. Um diese Erweiterungen nutzen zu können, muß die Document Type Definition des WML-Dokuments verändert

werden: `<!DOCTYPE wml PUBLIC „-//PHONE.COM//DTD WML 1.1//EN“ „http://www.phone.com/dtd/wml11.dtd“>`

Einige Ericsson-Modelle sowie viele der aktuellen Nokia-Telefone unterstützen den Download von Visitenkarten (vCards), die man im internen Adreßbuch ablegen kann, Ericsson außerdem den Download von Klingeltönen und den Versand von Kurznachrichten oder E-mail über einen `mailto:` Link.

Nicht nur aufgrund anderer Displaygrößen präsentieren verschiedene Telefon-Modelle WML-Dokumente sehr unterschiedlich. Nicht alle bieten die Möglichkeit in Grafiken, die nicht vollständig angezeigt werden können, zu scrollen. Auswahllisten und Eingabefelder werden manchmal als eigenes Menü, manchmal im Text der aktuellen Card angezeigt.

Ein weiterer, sehr wichtiger Punkt ist die Speicherkapazität der Telefone. Ist ein WML-Deck größer als 1.397 Byte, so friert der Browser des Nokia 7110 ein. Der UP.Browser dagegen unterstützt standardmäßig 1.492 Byte, Ericsson sogar 3.000 Byte (R320) und 3.800 Byte (R380).

Für einen WAP-Entwickler ist es daher fast unmöglich, Dokumente zu erzeugen, die für jeden Telefon-Microbrowser optimiert sind (eine User-Agent-Erkennung ist über die PHP-Variable $HTTP_USER_AGENT möglich, mehr dazu in Seeboerger-Weichselbaum 2001, S,. 212-216). Der Entwickler muß vielmehr darum bemüht sein, seine Dokumente so zu erstellen, daß eventuelle Mängel in der Darstellung (z. B. nicht unterstützte Schriftformatierungen) den Informationsgehalt nicht schmälern.

Eine Liste der aktuellen WAP-Mobiltelefone mit z. T. recht detaillierten Informationen zu Darstellung, WML/WMLScript/WTAI-Unterstützung und technischen Daten bietet die WWW-Seite des Autors Michael Seeboerger-Weichselbaum (http://www.wmlguru.net).

4.3.2 WML-Browser und Simulatoren

WML-Dokumente werden von den normalen WWW-Browsern nicht angezeigt, mit Ausnahme von Opera 4. Will man sie dennoch vom PC aus betrachten, so bieten sich zwei Möglichkeiten: Erstens kann man einen speziellen WML-Browser (z. B. WinWAP) installieren. Zweitens gibt es Simulatoren (als Software oder online), die bestimmte Mobiltelefone nachahmen. Der Benutzer bekommt ein Bild des Telefons samt möglichst originalgetreuem Displayinhalt angezeigt.

Für den häufigen Gebrauch ist ein WML-Browser oder Opera die bessere Wahl. Simulatoren eignen sich eher, um damit WML-Dokumente in der Entwicklung auf verschiedenen Mobiltelefonen zu testen, ohne diese Telefone besorgen zu müssen.

Verzichten muß der Nutzer bei WML-Browsern und Simulatoren auf WTAI-Funktionen und meistens auf WMLScript-Unterstützung. Abhilfe schaffen hier nur die Simulatoren, die in die Developer-Software der Hersteller integriert sind (Nokia Mobile Internet Toolkit, Openwave SDK, Ericsson WapIDE).

Tabelle 2: Entwicklungstools für WAP-Angebote

Name	**Hersteller**	**WWW-Seite**	**Anmerkung**
Nokia Mobile Information Toolkit 3.0	Nokia	erhältlich nach Registrierung unter http://www.nokia.com	Unterstützt bereits WAP 2.0, verschiedene Telefon-Simulatoren verfügbar
Nokia WAP Toolkit 2.0	Nokia	erhältlich nach Registrierung unter http://www.nokia.com	Vorgänger des Mobile Information Toolkit
Openwave SDK, WAP Edition	Openwave Systems	http://developer.openwave.com	Simuliert den UP.Browser (u. a. in Form eines Siemens S45), unterstützt WML 1.3 und WMLScript 1.2
Ericsson WapIDE	Ericsson	Erhältlich nach Registrierung unter: http://www.ericsson.com/	Simuliert das Ericsson R320, bietet im „Server Toolset“ einen WML- und WMLScript-Compiler.

Eine Auflistung und Diskussion aller verfügbaren WML-Browser würde den Rahmen dieser Diplomarbeit sprengen, daher hier nur Informationen zu den wichtigsten.

Tabelle 3: Die wichtigsten WML-Browser

Name	**Hersteller**	**WWW-Seite**	**Anmerkung**
WinWAP	Slob-Trot	http://www.winwap.com	Bedienung mit Maus und Tastatur, wie ein normales PC-Programm
Opera	Opera	http://www.opera.com	Neben HTML-Dokumenten kann Opera 4 auch WML und XML-Dokumente darstellen.
Wappy	wappy.to	http://wappy.to	Online-WML-Browser, zeigt alle Cards eines Decks an, unterstützt jedoch keine Variablen in WML, umschalten in eine Gerätesimulation möglich
iobox-Surfer	iobox	http://www.iobox.de	Der iobox-Surfer unterstützt nur recht wenige WML-Tags

Das gleiche gilt für Mobiltelefonsimulatoren, deshalb auch nur eine Auswahl der wichtigsten. Die meisten der Online-Simulatoren (keine Installation auf dem Rechner nötig), sind für das Testen von WML-Seiten ungeeignet. Sie simulieren die Mobiltelefone nur unzureichend, sodaß nicht einmal richtig eingeschätzt werden kann, wieviel Text auf das Display des Telefons passt.

Tabelle 4: Die wichtigsten Mobiltelefonsimulatoren

Name	**Hersteller**	**WWW-Seite**	**Anmerkung**
Wireless Companion	o3sis Information Technology AG	http://www.yourwap.com	Kombinierte Ansicht von Telefonsimulation und HTML-Browser. Der Simulator wird durch Anklicken der Telefon-Tasten bedient.
Wapalizer	gelon.net	http://www.gelon.net	Online-Simulator, mit vielen verschiedenen Telefonmodellen. Hat allerdings viele Probleme bei der Darstellung und Bedienung (z. B. Eingabe- und Auswahlfelder)
WapTiger	WapTiger	http://www.waptiger.de	Simulation eines Nokia 7110. Telefontasten haben andere Funktionen zugewiesen bekommen, Darstellung und Bedienung von Eingabefeldern problematisch
WapJag	WapJag	http://www.wapjag.de/	Organizer-artige Oberfläche, Probleme bei Formularen („Absenden“-Link wird z. T. nicht dargestellt).

WML-Browser gibt es nicht nur für den PC, auch der in der letzten Zeit rasch gewachsene Markt der Organizer und PDAs (Personal Digital Assistants) wird bedient. Viele Besitzer dieser Geräte verfügen vielleicht zusätzlich über ein WAP-Mobiltelefon, auf dem PDA ist die Darstellung von und Navigation in WML-Seiten allerdings wesentlich komfortabler. Auf einem Palm IIIc beispielsweise steht ein Display von 160x160 Pixel zur Verfügung, das zudem noch drucksensitiv ist, d. h. angezeigte Links können durch ein Antippen mit einem Spezialstift aktiviert werden. Außerdem besitzen diese Browser meist eine Navigationsleiste, wie sie von PC-HTML-Browsern bekannt ist („Vor“, „Zurück“, „Startseite“, „Laden abbrechen“).

Tabelle 5: WML-Browser für PDAs

Name	Hersteller	WWW-Seite	Anmerkung
WinWAP	Slob-Trot	http://www.winwap.org	Pocket PC-Variante des PC-WinWAP-Browsers
Klondike	Apache Software	http://www. apachesoftware.com	Browser für Pocket PC, Handheld 2000 und Windows Personal Edition
WAPman	Edgematrix	http://www.wap.com.sg	Für Palm und kompatible PDAs, benutzt immer das Gateway des Herstellers
KBrowser	4thpass	http://www.4thpass.com	Für Palm und kompatible PDAs, außerdem eine Java-Variante, die die J2ME (Java 2 PlatformMicro Edition) benötigt
WiredAnywhere	IBM	http://www.ibm.com	Für Palm und kompatible PDAs, z. Zt. noch in der Entwicklung

Besonders im Windows CE-Bereich sowie bei dem Betriebssystem EPOC (z. B. in Psion Organizern) gibt es noch einige herstellerspezifische Browser: Compaq bietet einen speziell Browser für seine Aero-Reihe an, Hewlett-Packard für die Jornada-Serie.

4.3.3 Spezielle Endgeräte

Abgesehen von den oben genannten Geräten und Browsern gibt es noch einige Spezialfälle. Beispielsweise war von Siemens ein WAP-fähiges Festnetztelefon angekündigt, oder von Becker ein Autoradio mit integriertem Telefon und WML-Browser (vgl. Immler/Kreinacke/Spallek 2000, S. 265/268). Leider sind keine aktuellen Nachrichten zu diesen Geräten zu bekommen. Ebenso ist das WAP-Gerät „Wapper“, das von der inzwischen nicht mehr existierenden Firma Tel@markt geplant war, verschwunden. Es sollte ein reines WAP-Endgerät sein, mit einer Displaygröße ähnlich der eines PDAs.

Eine interessante Kombination bietet die blue impact AG, Nachfolgerin der Tel@markt GmbH: den TellMEN. Durch einen spezielle Aufsatz wird aus einem Handspring-PDA (Visor) ein GSM-Telefon mit WAP-Terminal und Global Positioning System (GPS). Die besten Voraussetzungen für die komfortable Nutzung von Location Based Services.

5 Konzeption

5.1 Systemumgebung

5.1.1 Server

Die Erreichbarkeitsauskunft und der Verzeichnisdienst laufen auf dem Lehre-Server der Hochschule, einem LAMP-System. LAMP steht für Linux, Apache, mySQL, PHP und bezeichnet das Betriebssystem, den Webserver, das Datenbankmanagementsystem und die Skriptsprache, die auf dem System installiert sind.

Linux ist ein immer beliebteres Betriebssystem, insbesondere für Server. Der Apache-Webserver ist einer der am häufigsten eingesetzten Webserver überhaupt. mySQL ist ein Datenbankmanagementsystem für relationale Datenbanken. PHP ist eine Skriptsprache, die besonders gerne benutzt wird, um Inhalte dynamisch aus einer Datenbank zu erzeugen. Statt PHP könnte auch Perl zum Einsatz kommen. Ich habe mich für PHP entschieden, da es meiner Meinung nach für diesen Fall komfortabler ist. Sein Quelltext kann direkt in den der WML-Seiten eingebettet werden.

Vorteil eines LAMP-Systems ist, daß die gesamte benötigte Software kostenlos als Open Source (im Quelltext) zur Verfügung steht. Damit ist es preiswert und dennoch sehr leistungsfähig. So konnte ich, um die WAP-Services zu entwickeln, beispielsweise mySQL und PHP auf meinem persönlichen PC installieren, ohne daß mir Anschaffungs- oder Lizenzkosten entstanden.

5.1.2 Clients

Die auf die Anwendung zugreifenden Clients sind überwiegend Microbrowser von Mobiltelefonen. Desweiteren können Zugriffe erfolgen von WML-Browsern für PCs oder Organizern, sowie von Online- und Offline-Mobiltelefon-Simulatoren (vgl. 4.3 „WAP-Endgeräte“).

5.1.3 Datengrundlage

Die Datengrundlage für die Erreichbarkeitsauskunft und den Verzeichnisdienst bildet die bereits vorhandene Stundenplan-Datenbank für das Wintersemester 2001/2002. Diese mySQL-Datenbank verzeichnet Dozenten, Lehrveranstaltungen und Räume. Aus ihr werden das Vorlesungsverzeichnis, Stundenpläne für Dozenten und Belegungspläne für Räume erstellt. Außerdem basiert auf ihr das Dozenten- und Raumauskunftssystem.

Für die Erreichbarkeitsauskunft und den Verzeichnisdienst sind 4 Tabellen der Datenbank wichtig: „erfass“ enthält die Lehrveranstaltungen, „dozadr“ enthält die Dozenten, „lv“ kombiniert Dozent, Lehrveranstaltung, Zeitpunkt und Raum, „dozenten“ dient ei-

gentlich zur Verwaltung von Dozenten- bzw. Studentensemesterwochestunden, ist aber insofern interessant, da diese Tabelle die Dauer der Lehrveranstaltung enthält.

Tabelle 6 bis Tabelle 9 beschreiben die für die WAP-Services relevanten Felder aus diesen vier Tabellen.

Tabelle 6: Relevante Felder der Tabelle „erfass“

Feldname[5]	**Inhalt**	**Schlüssel**
Sigel	eindeutige Nummer der Veranstaltung	Primärschlüssel
titel	Titel der Veranstaltung	

Tabelle 7: Relevante Felder der Tabelle „dozadr“

Feldname	**Inhalt**	**Schlüssel**
nummer	eindeutige Nummer des Dozenten	Primärschlüssel
Name	Nachname des Dozenten	
Vorname	Vorname des Dozenten	
DozArt	Art des Dozenten (hauptamtlich, Lehrbeauftragter, Ruhestand)	
Strasse	Straße (Anschrift)	
Ort	Ort (Anschrift)	
PLZ	Postleitzahl (Anschrift)	
AdrArt	Art der Adresse (privat, dienstlich, keine)	
email	E-Mail-Adresse	
Telefon	Telefonnummer	
TelArt	Art der Telefonnummer (privat, dienstlich)	
TelDienst	dienstliche Telefonnummer	
TelMobil	Mobiltelefonnummer	
Fax	Faxnummer	

[5] Die Groß-/Kleinschreibung der Feldnamen entspricht der Schreibweise in der Datenbank.

Tabelle 8: Relevante Felder der Tabelle „lv“

Feldname	**Inhalt**	**Schlüssel**
Sigel	Nummer der Lehrveranstaltung	Fremdschlüssel aus „erfass“ bildet mit Feld „nummer“ den Primärschlüssel
nummer	Nummer des Dozenten	Fremdschlüssel aus „dozadr“ bildet mit Feld „Sigel“ den Primärschlüssel
Tag	Wochentag der Veranstaltung (mo, di, mi...)	
Zeit	Uhrzeit, zu der die Veranstaltung (0 = 8 Uhr, 1 = 9 Uhr, ...)	
Raum	Raum der Veranstaltung	

Tabelle 9: Relevante Felder der Tabelle „dozenten“

Feldname	**Inhalt**	**Schlüssel**
Sigel	Nummer der Lehrveranstaltung	Fremdschlüssel aus „erfass“ bildet mit Feld „Dozent“ den Primärschlüssel
Dozent	Nummer des Dozenten	Fremdschlüssel aus „dozadr“ bildet mit Feld „Sigel“ den Primärschlüssel
Durchführung	Dauer der Lehrveranstaltung (in Stunden)	

Da es noch einige Informationen gibt, die für den Nutzer interessant sind, war es nötig, die Stundenplan-Datenbank durch eine weitere Datenbank zu ergänzen. Diese zweite Datenbank hat kein Thema, sondern ist eine reine Sammlung ergänzender Tabellen: „einmalig“ enthält alle einmaligen Lehrveranstaltungen, da sie nicht in der eigentlichen Stundenplan-Datenbank stehen. In „abwesenheit“ stehen Einträge, von wann bis wann ein Dozent außerplanmäßig abwesend ist. „personen“ habe ich von meiner Kommilitonin Astrid Knoll übernommen. Sie entwickelt im Rahmen ihrer Diplomarbeit ein Raumauskunftssystem. „personen“ enthält die Raumnummern der Dozenten-Büros.

Tabelle 10 bis Tabelle 12 listen alle Ergänzungstabellen und ihre relevanten Felder auf.

Tabelle 10: Relevante Felder der Tabelle „einmalig“

Feldname	**Inhalt**	**Schlüssel**
Sigel	eindeutige Nummer für jede Veranstaltung	Primärschlüssel
datum	Datum der Veranstaltung (YYYY-MM-TT)	
tag	Wochentag der Veranstaltung (mo, di, ...)	
raum	Raum der Veranstaltung	
zeit	Zeit der Veranstaltung (hh:mm-hh:mm)	
dozentname	Name des Dozenten (kann auch externe Person sein)	
veranstaltung	Titel der Veranstaltung	

Tabelle 11: Relevante Felder der Tabelle „abwesenheit“

Feldname	**Inhalt**	**Schlüssel**
dozenten_nummer	Nummer des Dozenten	Fremdschlüssel aus „dozadr“
beginn	Unix-Timestamp des Abwesenheitsbeginns	
ende	Unix-Timestamp des Abwesenheitsendes	
grund	optionaler Eintrag des Abwesenheitsgrundes	

Tabelle 12: Relevante Felder der Tabelle „personen“

Feldname	**Inhalt**	**Schlüssel**
Nachname	Nachname des Dozenten	
email	E-Mail-Kennung ohne Angabe der Domain	
RaumID	Nummer des Büros	
Website	Website eines Dozenten	

Da die Datengrundlage aus verschiedenen Quellen stammt, ist sie leider sehr uneinheitlich. Die Verknüpfung von Tabellenfeldern für eine Datenbankanfrage, kann daher nicht immer über Schlüsselfelder geschehen, in drei Fällen ist ein Zeichenkettenvergleich über den SQL[6]-Ausdruck „LIKE“ nötig (vgl. Abbildung 10).

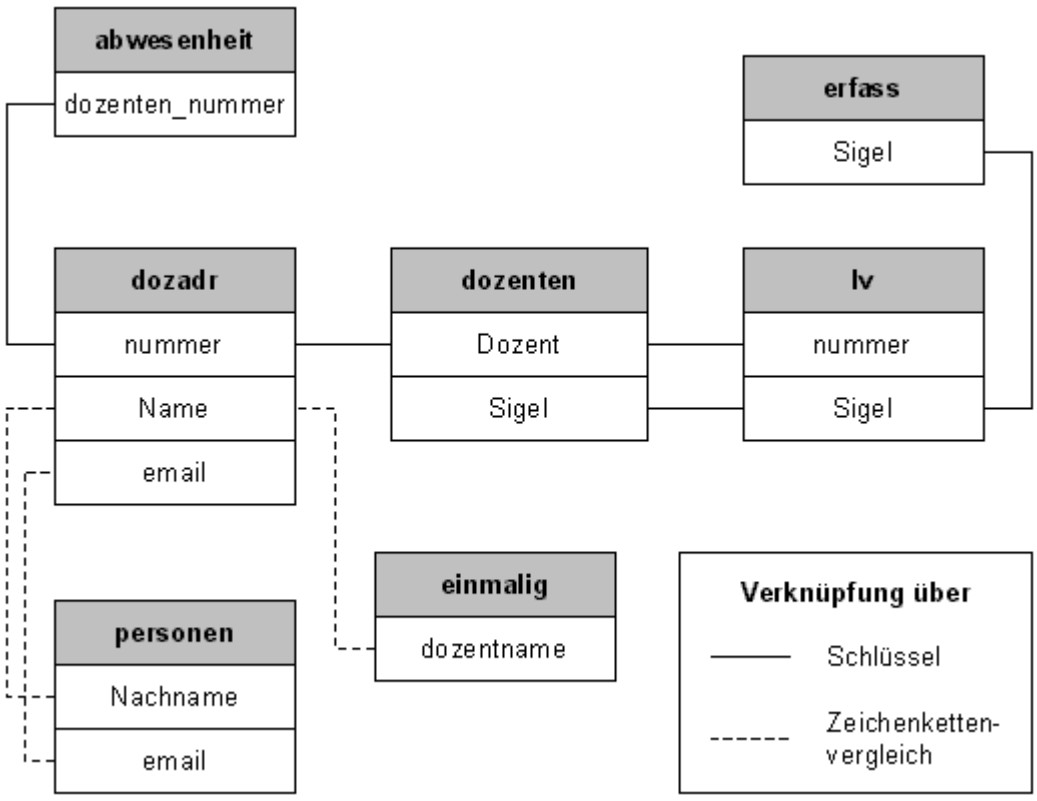

Abbildung 10: Verknüpfungen zwischen den Tabellenfeldern

[6] SQL (Structured Query Language) ist eine standardisierte Sprache zur Abfrage von relationalen Datenbanken.

5.2 Benutzungsoberfläche

Die Konzeption der Benutzungsoberfläche einer WAP-Anwendung stellt eine besondere Herausforderung dar. Das Ergebnis mag einfach aussehen, bedarf aber einiger Vorüberlegungen. Im Vergleich zur HTML-Oberflächengestaltung gibt es bei WML-Angeboten einige Einschränkungen.

Zunächst einmal sind die Voraussetzungen der Endgeräte, vor allem Mobiltelefone, zu beachten (vgl. auch Kapitel 4.3.1):

- Ein WML-Deck samt eventuell zugehöriger Bilder sollte nicht größer als 1.397 Byte sein. Dies entspricht der Speicherkapazität des Nokia 7110. Das Nokia 7110 ist das Gerät mit dem kleinsten Speicher. Als das erste WAP-fähige Telefon ist es sehr weit verbreitet und sollte daher als Maßstab dienen.
- Nicht jedes Endgerät unterstützt alle WML-Tags. Nicht immer werden verschiedene Schriftgrade und -schnitte unterstützt, oder Tabellen können nur einspaltig dargestellt werden. Der Informationsgehalt eines WML-Decks sollte nicht geschmälert werden, wenn nicht alle Formatierungen unterstützt werden.
- Die Eingabe größerer Textmengen auf einer Telefontastatur ist umständlich. Im Vergleich zu einer Computertastatur muß eine Taste bis zu siebenmal so oft gedrückt werden, um den gleichen Buchstaben zu erhalten.
- Zur Zeit unterstützen erst einige Mobiltelefone schnellere Übertragungsraten als 9.600 Bit/s. Dem Benutzer sollten nicht zu lange und häufige Wartezeiten durch das Herunterladen von WML-Decks zugemutet werden. Hier sollte die Möglichkeit genutzt werden, dem Benutzer in einem Deck mehr Informationen zu liefern als er eigentlich angefordert hat, die er aber wahrscheinlich anfordern wird.

Zusätzlich hat der Nutzer an ein WAP-Angebot andere Anforderungen als an ein WWW-Angebot:

- Er will schnell an Informationen gelangen. WAP ist kein Medium um zu browsen, sich also zwischen den Angeboten hin- und her zu bewegen, nicht unbedingt mit einem konkreten Ziel, sondern zu Unterhaltungszwecken. Ein WAP-Angebot muß so strukturiert sein, daß der Nutzer schnell zum gewünschten Ziel navigieren kann.
- Der Benutzer will eine angemessene Leistung für sein Geld erhalten. In (HS)CSD-Netzen zahlt er Gebühren für jede Minute, die er ein WAP-Angebot nutzt. Zur Zeit betragen diese Gebühren circa € 0,20 (0,39 Pf) pro Minute. Nur bei der Nutzung von GPRS kann die Abrechnung nach Datentransfer erfolgen. Nur wenn der Benutzer das Gefühl hat, daß das Preis-Leistungs-Verhältnis stimmt, wird er das Angebot wieder nutzen.

Als Umsteiger aus der WWW-Welt in die WAP-Welt sollte man sich immer vor Augen halten, daß WML nicht HTML ist, WAP nicht das World Wide Web.

Die ästhetische Gestaltung eines Angebotes ist zweitrangig. Das Design sollte zweckmäßig sein und den User bei der Benutzung unterstützen. Verzierende Elemente brauchen Bandbreite und haben kaum Nutzen für den Anwender. Ein Beispiel: Unterstreichungen eignen sich zwar gut, bestimmte Worte von anderen Worten abzuheben, der Benutzer wird sie jedoch auf den ersten Blick für Links halten, da er im WML-Browser nicht wie in einem HTML-Browser die Möglichkeit hat, unterstrichene Links von unterstrichenem Text durch die Farbe zu unterscheiden.

Eine einheitliche Darstellung auf allen Endgeräten ist nicht erstrebenswert. Die WML-Spezifikationen sagen ausdrücklich, daß den Herstellern Freiräume gelassen werden, um die Darstellung im Microbrowser der Darstellung der Telefon-Bedienelement anzugleichen: „WAP enables the creation of Man Machine Interfaces (MMIs) with maximum flexibility and ability for a vendor to enhance the user experience. In this way vendors can provide distinct user interfaces. For example, when the WAP Forum specifies user-interface elements, they do so in an abstract manner to allow for differentiation of implementations." (WAP Forum 2001b, S. 11). Letztlich profitiert hiervon der Benutzer: Er kennt die Benutzungsoberfläche seines Telefons und findet die gleichen Bedienelemente in WAP-Angeboten (z. B. Texteingabefelder, Buttons). Die Darstellung wird gewissermaßen automatisch an seine Gewohnheiten angepasst.

WML-Designer müssen sich im Prinzip den gleichen Herausforderungen stellen wie Web-Designer aus den ersten Jahren des World Wide Webs: langsame Verbindungen, hohe Gebühren und Browser, die wenig Gestaltungselemente darstellen konnten. Der Informationsgehalt ist höher einzuschätzen als die ästhetische Gestaltung.

5.2.1 Eingabe der Suchparameter

Die Eingabe der Suchparameter ist die wichtigste Stelle der Benutzungsoberfläche. An ihr interagiert der Nutzer mit dem WAP-Dienst. Diese Schnittstelle sollte daher für ihn leicht bedienbar und transparent in ihrer Struktur sein.

Taylor, M., Hosking, I, Brazier, D. (2000) unterscheiden zwischen zwei verschiedenen Navigationsmodellen für Eingabemasken: die Formular-basierte Navigation und die Frage-Antwort-Navigation. Beim Formularmodell kann der Benutzer auf der gleichen Anzeigeseite verschiedene Felder ansteuern, um dort Angaben zu machen oder zu verändern. Dagegen verlangt das Frage-Antwort-Modell von ihm in einer bestimmten Reihenfolge einzelne Angaben. Das Formularmodell eignet sich, wenn der Nutzer immer nur einige Angaben macht, beispielsweise bestimmte Einstellungen an seinem Telefon ändert. Würde er die Einstellungen nach dem Frage-Antwort-Modell präsentiert bekommen, müsste er durch Einstellungen blättern, die er gar nicht ändern will. Das Frage-Antwort-Modell ist geeignet, wenn ein Vorgang immer den gleichen Ablauf hat und dabei immer die gleichen Angaben benötigt. Ein Beispiel ist das Versenden einer Kurznachricht: Zuerst wird der Text geschrieben, dann der Empfänger bestimmt und darauf die Nachricht gesendet.

Ich habe mich für eine Frage-Antwort-Navigation entschieden, da die Auskunft immer die gleichen Angaben benötigt:

- Name des Dozenten
- gewünschter Zeitpunkt

Allerdings gibt es eine Abkürzung: Wenn ein Nutzer nur an den Kontaktdaten eines Dozenten interessiert ist, kann er die Angabe eines Zeitpunktes überspringen.

5.2.1.1 Dozent

Der Benutzer wählt einen gewünschten Dozenten über dessen Nachnamen aus. Den Dozenten sind in der Datenbank zwar auch Nummern zugeordnet, jedoch werden diese im normalen Hochschulbetrieb nicht benutzt, daher eignen sie sich nicht für eine Suche.

Diese Auswahl könnte man auf zwei Arten realisieren: Einerseits könnte man dem Benutzer eine hierarchische Struktur anbieten, durch die er sich hindurchklickt, andererseits kann man von ihm eine Eingabe in einem Textfeld verlangen.

Bei einer hierarchischen Struktur würde der Benutzer zuerst den Anfangsbuchstaben auswählen, dann den Dozenten. In der Datenbank sind 124 Dozenten verzeichnet mit 22 verschiedenen Anfangsbuchstaben. Es wäre unmöglich, die hierarchische Struktur in nur einem WML-Deck zu realisieren, da allein eine Liste mit verlinkten Nachnamen in Bytecode schon 2357 Byte lang ist. Also würde der Benutzer den Anfangsbuchstaben auswählen, dann warten, bis die Namensliste geladen wurde, dann den Zeitpunkt angeben und wieder warten, bis er vom Server das Ergebnis seiner Anfrage erhält. Im ungünstigsten Fall, dem alphabetisch letzten Namen mit Anfangsbuchstaben „S“, muß er 33 mal eine Taste drücken: 16 mal, um die Buchstaben entlang bis „S“ zu springen, dann noch 17 mal, um bis zum letzten Namen in der Liste zu springen.

Die hierarchische Variante ist meiner Meinung nach in diesem Fall für den Benutzer zu unkomfortabel. Erstens muß er, bis er eine Auskunft erhält, zweimal auf das Laden eines Dokumentes warten, zweitens muß er bis zu 33 mal eine Taste drücken. Daher habe ich der Benutzereingabe den Vorzug gegeben. Dabei gibt der Benutzer über das Tastenfeld des Telefons den Namen an, wie beim Verfassen einer Kurznachricht. Im ungünstigsten Fall muß der Benutzer 29 mal eine Taste drücken („Feistritzer“, 3 mal für „f“, 2 mal für „e“, 3 mal für „i“, ...). Nach der Eingabe des Namens kann der Benutzer sofort mit der Auswahl des Zeitpunktes fortfahren. Erst danach wartet er, bis er das Ergebnis seiner Anfrage erhält. Im Gegensatz zur hierarchischen Variante wird eine Wartepause eingespart.

Natürlich ist es nicht immer notwendig, den gesamten Nachnamen einzugeben. Abbildung 11 zeigt, nach der Eingabe von wievielen Buchstaben ein Nachname sich von den vor und nach ihm im Alphabet stehenden Namen unterscheidet (z. B. unterscheiden sich „Maier“ von „Müller“ nach 2 Buchstaben).

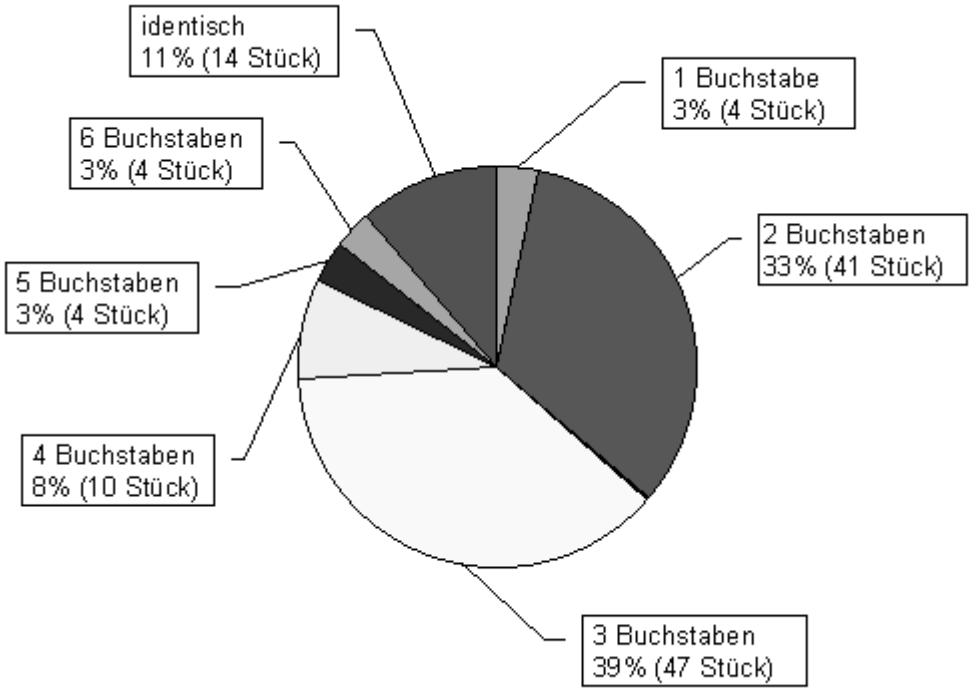

Abbildung 11: Anzahl der nötigen Buchstaben um einen Namen von alphabetisch nahestehenden Namen zu unterscheiden

Bereits mit der Eingabe von 3 Buchstaben erzielt die Suche zu 75 % genau einen Treffer. In den restlichen 25% der Fälle bekäme der Benutzer eine Auswahlliste mit den gefundenen Nachnamen angezeigt.

In der optimierten Variante der Namenseingabe wird der Benutzer aufgefordert, nur die Anfangsbuchstaben des Namens anzugeben. Diese werden dann zusammen mit dem angegebenen Zeitpunkt an den Server geschickt.

Falls der Benutzer keinen Dozenten angibt, bekommt er eine Fehlermeldung.

5.2.1.2 Zeitpunkt

Um die Auswahl des Zeitpunktes für den Benutzer so komfortabel wie möglich zu gestalten, muß man sich im klaren darüber sein, welche Zeitpunkte ihn interessieren. Häufig wird dies, wie im eingangs beschriebenen Szenario, die nahe Zukunft sein: jetzt, in einigen Stunden, morgen.

Der Benutzer bekommt natürlich die Möglichkeit, ein bestimmtes Datum und eine bestimmte Uhrzeit einzugeben. Viel wichtiger ist aber die Schnellauswahl Jetzt/Morgen zu dieser Zeit/in x Stunden (Benutzereingabe der Anzahl).

Zusätzlich kann der Benutzer die Zeitpunkt-Auswahl überspringen, um nur die Adressdaten eines Dozenten anzufordern.

5.2.2 Anzeige der Erreichbarkeit

Die Grundinformation, die ein Benutzer erhalten will ist „Ja, die Person ist an der Hochschule erreichbar.“ bzw. „Nein, die Person ist nicht erreichbar.“. Ist die Person nicht erreichbar, wird den Benutzer im Normalfall interessieren, wann die Person denn dann erreichbar ist. Und wenn die Person erreichbar ist, interessiert ihn vielleicht, wo sie sich aufhält, oder wieviel Zeit sie voraussichtlich zur Verfügung hat. Deshalb liefert der WAP-Dienst nicht die reine Ja/Nein-Antwort, sondern Informationen, anhand derer der Benutzer abschätzen kann, ob er die Person erreichen kann, und wenn ja wo und wann.

Was die Erreichbarkeit betrifft, können anhand der Datengrundlage folgende Fälle unterschieden werden:

1. Der Dozent hält dieses Semester keine Lehrveranstaltungen.
2. Der Dozent hält an diesem Tag keine Lehrveranstaltungen.
3. Der Dozent wird an diesem Tag eine Lehrveranstaltung halten.
4. Der Dozent hält gerade eine Lehrveranstaltung.
5. Der Dozent hat eine Lehrveranstaltung gehalten und wird eine/mehrere weitere halten.
6. Der Dozent hat eine Lehrveranstaltung gehalten und wird keine weitere halten.

Darauf basierend können die in Tabelle 13 dargestellten Annahmen, Handlungen und Informationsbedarfe des Benutzers abgeleitet werden, auf die die Erreichbarkeitsauskunft eine Antwort gibt.

Tabelle 13: Annahmen, Handlungen und Fragen des Benutzers und Antwort der Auskunft

Fall	**Annahme des Benutzers**	**mögliche Handlung des Benutzers**	**Frage des Benutzers**	**Antwort auf die Frage durch:**
1, 2	Der Dozent ist nicht an der Hochschule erreichbar.	Den Dozenten anderweitig erreichen.	Wie kann ich ihn sonst erreichen?	Verweis auf Kontaktdaten
		Es an einem anderen Tag probieren.	Ist er morgen/an anderem Tag erreichbar?	Neue Anfrage an die Auskunft.
3	Der Dozent wird bis zur Lehrveranstaltung an die Hochschule kommen.	Versuchen, den Dozenten vor der Lehrveranstaltung zu erreichen.	Wann beginnt die Lehrveranstaltung? Wo findet sie statt?	Erreichbarkeitsinformation

4	Der Dozent ist in der Lehrveranstaltung.	Versuchen, den Dozenten nach der Lehrveranstaltung bzw. in einer eventuellen Pause zu erreichen.	Wo findet die Lehrveranstaltung statt? Wie lange dauert sie?	Erreichbarkeitsinformation
5	Je nach Länge der Pause ist der Dozent an der Hochschule oder auch nicht, wird aber direkt nach bzw. vor der Lehrveranstaltung da sein.	Den Dozenten in der Pause zwischen den Lehrveranstaltungen erreichen, bzw. anderweitig.	Wann und wo endet die letzte Vorlesung, wann und wo beginnt die nächste?	Erreichbarkeitsinformation
			Wie ist der Dozent anderweitig zu erreichen?	Verweis auf Kontaktdaten
6	Der Dozent wird nur noch an der Hochschule sein, wenn die Veranstaltung gerade eben endete.	Den Dozenten noch an der Hochschule oder sonst anderweitig erreichen.	Wann und wo endet die letzte Vorlesung?	Erreichbarkeitsinformation
			Wie ist der Dozent anderweitig zu erreichen?	Verweis auf Kontaktdaten

Der einfachen Verständlichkeit halber erfolgt die Ausgabe der Erreichbarkeitsauskunft in einem natürlichsprachlichen Satz, ergänzt durch das Datum des Tages, für den die Aussage gilt: „XY hält im Moment die Lehrveranstaltung „YZ“ in Raum w012. Die Lehrveranstaltung endet voraussichtlich um 15:00 Uhr. (2001-10-29)“. Danach folgt ein Link zu den Kontaktdaten des Dozenten.

5.2.3 Anzeige der Kontaktdaten

Die Kontaktdaten liefern dem Nutzer eine Art Visitenkarte der gesuchten Person. Je nachdem welche Einträge in der Stundenplandatenbank eingetragen sind, bekommt der Benutzer folgendes angezeigt:

- Titel, Vorname, Nachname
- dienstliche Telefonnummer, private Telefonnummer, Mobiltelefonnummer, Faxnummer
- Dienstanschrift, Privatanschrift
- E-mail-Adresse, Website
- Zimmernummer des Büros

Zusätzlich kann der Benutzer eine Visitenkarte (nach dem vCard-Standard, Version 2.1, vgl. versit Consortium 1996) herunterladen, sofern es sein Endgerät unterstützt. Der vCard-Standard wird u. a. unterstützt vom Ericsson R380e, fast allen Nokia-Modellen, dem Trium Eclipse und den Siemens-Modellen SL 42 und SL 45 (jeweils laut Herstellerangaben). Außerdem wird das vCard-Format von der gängigen PC-Kontaktverwaltungs-Software unterstützt (z. B. Microsoft Outlook (Express), Netscape Messenger (mit Einschränkungen)), sowie den meisten PDAs (z. B. Palm, Handspring).

5.2.4 Daraus resultierende Deck- und Card-Struktur

Die vorhergehend ausgeführten Punkte werden in 3 WML-Decks umgesetzt: das Eingabedeck, das Auskunftsdeck und das Kontaktdaten-Deck. Wichtig ist, daß dem User die Möglichkeit gegeben wird, über Links sich auch wieder rückwärts durch seine Eingaben auf verschiedenen Cards zu bewegen, da die „Zurück"-Funktion in den WML-Browsern der Telefone meistens etwas versteckt ist.

5.2.4.1 Eingabedeck

Das Eingabedeck enthält 5 Cards:

- **Begrüßung**:
  "Willkommen bei den WAP-Services der HdM Stuttgart!"
  Falls er nicht weiterklickt, wird der Benutzer nach einigen Sekunden automatisch zur nächsten Card weitergeleitet.
- **Kurze Erklärung der Funktionsweise der Auskunft**:
  "Sie geben den Namen ein, wählen den Zeitpunkt aus, und diese Auskunft sagt Ihnen, ob die Person an der Hochschule erreichbar ist."
  Der Benutzer wird informiert, welche Eingaben von ihm erwartet werden, sodaß bei der eigentlichen Eingabe keine Erklärung mehr benötigt wird.
- **Eingabe des Namens(anfangs)**:
  Der Benutzer gibt die ersten Buchstaben des Namens ein, klickt auf „Weiter" und kommt zur Auswahl des Zeitpunkts. Die von ihm eingegebenen Buchstaben werden in einer WML-Variablen gespeichert. Diese Variable wird bei der Anfrage immer übergeben.
- **Auswahl des Zeitpunkts**:
  "Für welchen Zeitpunkt wollen Sie Informationen über "[Hier werden die in der Variablen gespeicherten Buchstaben ausgelesen und angezeigt]..."?"
  Der Benutzer bekommt verschiedene Auswahlmöglichkeiten präsentiert. Je nachdem, welche er wählt, werden verschiedene WML-Variablen gesetzt und mit der bereits gesetzten Variablen an den Server übergeben:
  - „Nur Kontaktdaten":
    In diesem Fall werden ihm nur die Kontaktdaten, nicht die Erreichbarkeitsin-

formationen ausgegeben. Die Variable für den Anfragetyp wird auf „ad“(resse) gesetzt und an den Server übergeben.

- „Jetzt“:
  Der Anfrage-Typ auf „plus“ gesetzt, die Variable für die hinzuzuzählenden Stunden auf 0 und zusammen mit der aktuellen Zeit an den Server übergeben. Der Anfragetyp „plus“ ist eigentlich gedacht für die Auswahlmöglichkeit „in x Stunden“. Hiermit wird bei der Programmierung eine weitere Fallunterscheidung umgangen, die ein eigener Abfragetyp „Jetzt“ erfordern würde.
- „Morgen, diese Zeit“:
  Bei der Auswahl wird hier der Anfragetyp ebenfalls auf „plus“ gesetzt, die hinzuzuzählenden Stunden auf „24“ und zusammen mit der aktuellen Zeit an den Server übergeben.
- „in x Stunden“:
  In einem zusätzlichen Eingabefeld kann der Benutzer angeben, wieviele Stunden es genau sein sollen. Die Anzahl der Stunden, der Anfragetyp „plus“ und die aktuelle Zeit werden an den Server übergeben.
- „genauen Zeitpunkt eingeben“:
  Falls keine der Schnellwahlen den Benutzer zufriedenstellt, wird er über diesen Link auf die letzte Card des Decks verwiesen. Hätte er eine der vorhergehenden Auswahlmöglichkeiten genutzt, würde er von der letzten Card nichts bemerken.

- **Eingabe des genauen Zeitpunkts**
  Auf dieser Card gibt der Benutzer (über die Telefontastatur) den Tag ein, wählt aus einer Liste den Monat aus (Oktober 2001 - März 2002) und tippt die gewünschte Stunde ein. Diese drei Werte werden in drei Variablen gespeichert, die an den Server übergeben werden. Das vom Benutzer eingegebene Datum und Uhrzeit werden auf ihre Richtigkeit geprüft, bevor das auswertende Programm sie weiterverarbeitet.

5.2.4.2 Auskunftsdeck

Die Anzahl der im Auskunftsdeck enthaltenen Cards variiert. Gibt es für den vom Benutzer eingegebenen Namensanfang nur eine Übereinstimmung, enthält es eine einzige Card mit den Erreichbarkeitsinformationen und einem Verweise auf die Adressdaten der betreffenden Person.

Werden mehrere Übereinstimmungen für den Namensanfang gefunden, ist die erste Card des Auskunftsdecks eine Liste der gefundenen Personen. Danach folgen pro Person eine Card mit den Erreichbarkeitsinformationen. Wählt der Benutzer eine der Personen aus, wird ihm die entsprechende Auskunft gegeben. Die restlichen Cards bleiben für ihn verborgen. Das Deck kann dadurch zwar der Größenbegrenzung von 1397 Byte nahekommen, für den Nutzer hat es jedoch den Vorteil, daß er nach der Auswahl des passenden Suchtreffers nicht mehr auf die Auskunft warten muß. Erst wenn das Deck 1397 Byte übersteigen würde, muß der Nutzer nach der Auswahl der richtigen Person auf eine Auskunft warten (Dieser Fall tritt nur bei einer Suche nach „B“, „H“, „S“, „Sc“ und „Sch“ auf).

5.2.4.3 Kontaktdaten-Deck

Das Kontaktdaten-Deck enthält eine Card mit den in der Datenbank gespeicherten Kontaktdaten der Person: Telefonnummern, Anschriften, E-mail-Adressen, Internet-Seite, Büronummer. Der Benutzer soll die Möglichkeit bekommen, angezeigte Telefonnummern gleich über das WTAI weiterverarbeiten zu können. Daher verweist jede Telefonnummer als Link auf jeweils eine weitere Card, die für diese Nummer die Funktionen „Wählen“ und „Abspeichern“ anbietet.

5.3 Einschränkungen

5.3.1 Wahrung der Kompatibilität zu verschiedenen WML-Browsern

Aus der Vielfalt der WAP-Endgeräte und ihrer WML-Browser ergeben sich einige Einschränkungen, die es zu beachten gilt. Zum Teil wurden sie bereits oder werden später noch einmal erwähnt, daher hier die zusammenfassende Übersicht.

5.3.1.1 Dateigröße

Das Mobiltelefon Nokia 7110 setzt hier den Standard. Seine Speicherkapazität beträgt 1397 Byte. Ein WML-Deck sollte, in Bytecode umgewandelt, diese Grenze nicht überschreiten. Bei größeren Dateien friert der Browser des Nokia 7110 ein. Damit er wieder funktioniert, muß der User das Telefon aus- und wieder einschalten. Neuere Modelle haben einen größeren Speicher und ihr Browser ist fehlertoleranter. Ist eine Datei zu groß für den Speicher, zeigt beispielsweise das Siemens S35i nur den Teil des WML-Decks an, der in den Speicher passt.

5.3.1.2 WML-Tags

Nicht alle Mobiltelefone unterstützen alle WML-Tags. Das soll aber nicht heißen, daß man auf alle unter Umständen nicht unterstützten Tags verzichten muß, und daß ein WML-Dokument nur aus unformatiertem Text und einem Link pro Card bestehen kann. Man sollte sich nur bewußt sein, welche Auswirkungen es hat, wenn bestimmte Tags nicht richtig interpretiert werden. Wird das Dokument nur etwas unübersichtlicher oder verliert es an Informationsgehalt?

Besonders zu beachten sind:

- Textformatierungen:
  Nokia 7110 und 6210 unterstützen weder fett (`<b>`), kursiv (`<i>`), unterstrichen (`<u>`), groß (`<big>`), klein (`<small>`), das Siemens S35 dagegen unterstützt fett, kursiv und teilweise große Schrift (wird fett formatiert). Das Sony CMD Z-5 dagegen unterstützt alle genannten Formatierungen.
- Softkey-Belegungen:
  Die meisten Telefone besitzen zwei Softkeys, jedoch können nicht immer beide mit

Funktionen belegt werden. Einer ist oft für Telefon-Funktionen belegt, die für ihn vorgesehene Funktion findet sich meistens in dem Menü, das aufklappt, wenn der Softkey gedrückt wird. Wichtige Links sollten daher möglichst nicht nur auf Softkeys gelegt werden.

- Tabellen:
  Das Nokia 7110 kann Tabellen nur einspaltig darstellen, die restlichen Telefone haben meistens Probleme, wenn die Tabelle breiter als das Display ist.

Tabelle 14 zeigt exemplarisch, wie ein WML-Dokument optimiert werden kann, damit es keine Informationen durch die Nicht-Interpretation von WML-Tags verliert.

Tabelle 14: Vermeidung von Informationsverlust durch Optimierung der verwendeten WML-Tags

	Ausgangs-Dokument	**Auf Kompatibilität optimiert**
Problem und Lösung	Information geht verloren (steigende/fallende Kurse), der Benutzer muß Links im Softkey-Menü suchen.	Information bleibt erhalten, Textformatierung hat unterstützenden Charakter, Softkey-Links wurden in den Text verlagert, statt dessen wird ein „Zurück“-Link angeboten.
Browser unterstützt alle verwendeten WML-Tags		
Browser unterstützt nur wenige WML-Tags		
Quellcode	`<card>`	`<card>`

	<do type="option" label="ok"> <go href="#detailauswahl"/> </do> <do type="accept" label="•/Stück"> <go href="#ansichtstueck"/> </do> <p> <big>Ihr Depot Anteilsbestände</big> Fonds: •745,30 Aktien: •1.045,21 <small> Erklärung: ... </small> </card>	<do type="prev" label="Zurück"> <go href="#anfang"/> </do> <big>Ihr Depot Anteilsbestände</big> Fonds: •745,30 (+) Details Aktien: 1.045,21 Details Umschalten: ... </card>

Für die im Rahmen dieser Diplomarbeit entwickelten WAP-Dienste ergibt sich folgendes:

- Textformatierungen dienen nur der Übersicht und haben keinen informativen Charakter. Unterstreichungen werden vermieden, damit sie nicht mit Links verwechselt werden können.
- Softkey-Links werden definiert. Falls zwei Softkeys belegt werden, werden die Links zusätzlich im Text angeboten.
- Wichtige Links (wie das Absenden eines Formulars) werden durch ihre Beschriftung zusätzlich hervorgehoben: [..OK..], ebenso werden Eingabefelder mit einem Pfeil als Prompt betont: -> .

5.3.1.3 Bilder

Bilder sollten die Anzeigegröße eines Endgerätes nicht übersteigen. Die wenigsten Geräte bieten eine horizontale Scroll-Funktion an und schneiden die Grafik einfach ab.

Desweiteren rauben Bilder Speicherplatz. Für ein Bild, dessen Größe etwa einem halben Display entspricht (50x100 Pixel), werden 654 Byte Speicher benötigt. Der Platz für WML-Dokumente reduziert sich um diesen Anteil.

Eine Alternative bieten lokal in den Endgeräten gespeicherte Bilder. Allerdings besitzen sie nicht alle Geräte, sie sind recht klein und können natürlich kein Firmen-Logo ersetzen.

Das einzige Bild, das für die WAP-Services der HdM in Frage kommen würde, wäre das Signet (Logo) der Hochschule. Um erkennbar zu sein, benötigt es allerdings eine

Höhe von 40 Pixel und eine Breite von 140 Pixel, was es unbrauchbar für die Darstellung auf Telefondisplays macht.

5.3.1.4 WMLScript

Alle Mobiltelefone unterstützen WMLScript, aber keiner der PC- bzw. Online-WML-Browser, ausgenommen der Simulatoren in den Entwicklungsumgebungen. Wichtige Funktionen sollten daher nicht über WMLScript realisiert werden.

Es hätte sich angeboten, eine Datumseingabe mit Hilfe einer WMLScript-Funktion auf ihre Gültigkeit zu prüfen. Wie bei ähnlichen JavaScript-Funktionen übernimmt die WML-Script-Funktion zwangsweise das Absenden der Formulardaten an den Server. Nachdem die meisten WML-Browser für den PC WMLScript nicht unterstützen, muß auf diese Funktion verzichtet werden. Es wäre mit diesen Browsern nicht möglich gewesen, die zur Suche nötigen Daten an den Server zu senden. Die Datums-Überprüfung geschieht nun serverseitig im auswertenden PHP-Skript.

5.3.2 Zugriffsregelung

Zugriffsregelungen können aus zwei Arten geschehen. Man kann ein WAP-Intranet aufbauen, indem man entweder einen eigenen Einwahlserver betreibt oder zumindest ein eigenes Gateway auf seinem Webserver installiert, dessen Adresse nur den autorisierten Personen mitgeteilt wird. Obwohl die Ereignisauskunft und der Verzeichnisdienst größtenteil HdM-interne Personen (Dozenten, Verwaltung, Studenten) anspricht, ist diese Art der Zugriffsregelung nicht praktikabel. Erstens ist es mit hohem Installationsaufwand auf dem Server verbunden und der Einwahlserver würde wahrscheinlich weitere Telefonleitungen benötigen, was Geld kostet. Müsste jeder Interessierte sein Telefon zuerst neu konfigurieren, damit er Zugriff auf die WAP-Dienste hat, wäre das eine Hürde, die viele an der Erstnutzung dieser Dienste hindern würde.

Die zweite Variante der Zugriffseinschränkung liegt in WML. Mit Hilfe des Tags `<access domain="..." path="..."/>` kann bestimmt werden, aus welcher Domain und eventuell welchem Pfad auf die Cards eines Decks zugegriffen werden darf. Diese Möglichkeit greift bei den WAP-Dokumenten der Auskunft und des Verzeichnisdienstes nicht, da sie durch das Aufrufen eines PHP-Skripts erzeugt werden, das man auf diese Weise nicht vor ungewollten Zugriffen schützen kann.

5.3.3 Nutzung der WTAI-Bibliotheken

Der Benutzer hat die Möglichkeit, Telefon- bzw. Faxnummern, die ihm angezeigt werden, mit seinem Telefon weiter zu bearbeiten. Hierfür kann allerdings nur die öffentliche WTAI-Bibliothek benutzt werden, mit den drei Funktionen Anrufen, Abspeichern und DTMF-Töne senden. Alle anderen WTAI-Bibliotheken können nicht genutzt werden, da die HdM nicht von den Mobilfunkprovidern entsprechend zertifiziert wurde.

5.3.4 Aktualität und Pflege der Datengrundlage

Die Qualität einer Auskunft kann nur so gut sein, wie die Daten anhand derer sie Auskünfte erteilt. Aktualisierung und Pflege der Stundenplan-Datenbank unterstehen dem Stundenplanamt. Die ergänzenden Tabellen sind Auszüge anderer Datenbanken, die für die Auskunft und den Verzeichnisdienst nicht direkt zur Verfügung stehen. Sie können leider nur von Hand in regelmäßigen Abständen aktualisiert werden. Insofern können die Auskünfte nur ohne Gewähr erteilt werden. Der Benutzer kann aber davon ausgehen, daß der Großteil der erteilten Auskünfte stimmen dürfte, schließlich ist die Stundenplan-Datenbank für den Regelbetrieb der Hochschule nötig und die ergänzenden Angaben, wie beispielsweise Büronummern ändern sich nicht so häufig.

6 Durchgeführte Arbeiten

6.1 Zur Entwicklung verwendete Software

Zur Entwicklung der WAP-Services habe ich folgende Software benutzt:

- **PHPEd, Version 1.45 alpha**[7]
  PHPEd bezeichnet sich selbst als "Integrated Debugging Environment for php". Es diente mir zum Schreiben der PHP-Skripte. Hilfreich ist vor allem das Syntax-Highlighting, die Nachschlagefunktion für PHP-Befehle und -Funktionen, sowie die Möglichkeit, das PHP-Skript ablaufen zu lassen, ohne den Webserver starten zu müssen. Ursprünglich ist PHPEd für PHP 3 gedacht, es ist jedoch kein Problem, damit in PHP 4 zu programmieren.

- **Nokia WAP Toolkit 2.0** bzw. **Nokia Mobile Internet Toolkit 3.0**[8]
  Das WAP Toolkit und sein Nachfolger Mobile Internet Toolkit habe ich benutzt, um WML-Quelltext auf Fehler zu überprüfen, sowie um die Anzeige der WML-Dokumente auf den Nokia-Simulatoren zu testen (Nokia 7110, Nokia 6210, Nokia Blueprint Phone und zwei Simulatoren, die keinen realen Telefonmodellen entsprechen, sondern den Nokia WML-Browser simulieren).

- **Openwave SDK, WAP Edition 5.0**[9]
  Das Openwave SDK (Software Development Kit) dient wie die Nokia Toolkits zur Entwicklung von WAP-Anwendungen, allerdings wird hier der Openwave Browser (UP.Browser) simuliert. Das Openwave SDK hat einen noch größeren Funktionsumfang als die Nokia Toolkits. Ich habe es trotzdem nur wegen des Simulators für das Siemens S45 benutzt, da das Development Kit an sich noch sehr fehlerbehaftet ist und selten problemlos funktioniert.

- **YOURWAP.com Wireless Companion**[10]
  Der Wireless Companion ist eine Kombination aus WML- und HTML-Browser. In verschiedenen Versionen simuliert er unter anderem das Nokia 7110/6210, Siemens M/C/S35 und den IC35, Ericsson R320/R380s, Motorola Timeport P7389/P2288 und das Alcatel db view. Die Darstellung der WML-Dokumente erfolgt nicht immer originalgetreu, jedoch leistete der Wireless Companion gute Dienste bei der Simulation der Telefonbedienung, die sehr originalgetreu umgesetzt wurde.

---

[7] http://www.soysal.com/PHPEd

[8] http://forum.nokia.com/wapforum/main/toolkit

[9] http://developer.openwave.com/

[10] http://www.yourwap.com

- **WinWAP 3.0 PRO**[11]
  WinWAP ist ein WML-Browser für den PC. Er bietet keinerlei Telefon-Simulationen. Ich setzte ihn ein, um WML-Dokumente auf ihre Funktionsfähigkeit zu testen, da er sich schnell und einfach mit Maus und Tastatur bedienen lässt.

- **mySQL 3.23.33**[12]
  Es war nötig, das Datenbankmanagementsystem mySQL auf meinem Rechner zu installieren, damit ich lokal mit einer Kopie der Stundenplan-Datenbank arbeiten konnte. Außerdem erstellte ich damit die ergänzende Datenbank und ihre Tabellen. mySQL ist Open Source, d. h. es steht kostenlos im Quelltext zur Verfügung.

- **Xitami, v2.4d5**[13]
  Xitami ist ein leicht zu bedienender, als Open Source verfügbarer Webserver. Er war nötig, damit ich lokal PHP-Skripte ausführen konnte, um die von ihnen generierten Ausgaben in WML-Browsern und Simulatoren zu betrachten.

- **PHP 4.0**[14]
  Die Installation von PHP 4.0 war unabdingbar, um PHP-Skripte auf meinem Rechner ausführen zu lassen.

Zusätzlich zu der hier aufgeführten Software stand mir mein Siemens S35i (UP.Browser Version 4.1.16m) zur Verfügung, um WML-Dokumente zu testen. Allerdings wurden die Tests häufig durch ein überlastetes Gateway des Providers verhindert, da im Rahmen einer Werbeaktion jedem Kunden WAP-Freiminuten geschenkt worden waren.

Es existiert noch keine Entwicklungsumgebung, die WML und PHP integriert. Zur Zeit muß man noch mit verschiedenen Anwendungen parallel arbeiten: PHP-Skripte können beispielsweise in PHPEd ausgeführt werden, aber um das erzeugte WML-Dokument betrachten zu können, muß entweder die Ausgabe des Skriptes abgespeichert werden und in einem Simulator geöffnet, oder die Datei mit dem PHP-Skript muß im Webserver-Verzeichnis gespeichert werden, und wird dann unter der Serveradresse im Simulator geöffnet.

6.2 Vorbereitungen auf dem Webserver

Damit auf dem Lehre-Server des Fachbereichs 3 der HdM[15] die WAP-Dienste angeboten werden konnten, mußten zusätzliche MIME-Types auf dem Webserver eingetragen werden. MIME steht für Multipurpose Internet Mail Extensions und ist notwendig, damit Nicht-Text-Dateien (z. B. Bilder) im Internet übertragen werden können. Der MIME-Type einer Datei wird bei jeder Anforderung eines Clients vom Server mitge-

[11] http://www.slobtrot.com
[12] http://www.mysql.com
[13] http://www.imatix.com
[14] http://www.php.net

schickt. Wird der falsche MIME-Type gesendet, kann es sein, daß ein HTML-Dokument beispielsweise als Text dargestellt wird.

Tabelle 15: Für die WAP-Dienste zusätzlich notwendige MIME-Types

MIME-Typ	**Dateiendung**	**Beschreibung**
text/vnd.wap.wml	.wml	WML-Dokument
text/vnd.wap.wmlscript	.wmls bzw. .wmlscript	WMLScript-Datei
application/vnd.wap.wmlc	.wmlc	kompilierter WML-Code (nicht unbedingt nötig)
application/vnd.wap.wmlsc	.wmlsc	kompiliertes WMLScript-Skript (nicht unbedingt nötig)
image/vnd.wap.wbmp	.wbmp	Wireless Bitmap
text/x-vcard	.vcf	Visitenkarte im vCard-Format

Zuerst sollten die WAP-Dienste in meinem Home-Verzeichnis auf dem Lehre-Server liegen, dann stellte sich allerdings heraus, daß Mobiltelefontastaturen normalerweise das Zeichen „~“ (Tilde) nicht unterstützen, das Bestandteil der Adresse war. Jetzt erreicht man die WAP-Dienste über den Link http://v.hdm-stuttgart.de/wap.

6.3 Programmierung

Die in diesem Kapitel angeführten Quelltext-Beispiele wurden zum Teil vereinfacht, damit sie für den Leser übersichtlich bleiben. Die vollständigen Quelltext befinden sich auf dem beigefügten Datenträger.

6.3.1 Eingabedeck *index.wml*

Das Eingabedeck ist eine normale WML-Datei: *index.wml*. Zuerst dachte ich, es sei notwendig, sie dynamisch zu erzeugen, da sie in einem Formularfeld die aktuelle Uhrzeit übergeben sollte. Da viele Endgeräte keine Uhr oder einen Kalender besitzen, ist in WMLScript nicht wie in JavaScript eine Funktion definiert, die auf dem Client Uhrzeit- oder Datumsangaben ausliest. Das Eingabedeck hätte deshalb ein PHP-Skript sein müssen.

Bei dieser Zeitangabe kommt es jedoch nicht auf die Sekunde an, deshalb habe ich mich entschlossen, die aktuelle Zeit im auswertenden Skript zu generieren, sobald die Daten aus dem Eingabedeck übergeben wurden. Dadurch spare ich einerseits Ressourcen des

[15] http://v.hdm-stuttgart.de

Servers, da er ein PHP-Skript weniger auszuführen hat, und andererseits ist der WML-Quelltext lesbarer, wenn er keine PHP-Anteile enthält.

6.3.1.1 Doctype, Kopfbereich des Dokuments

```
<?xml version="1.0"?><!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML
1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <head>
    <meta name="author" content="Susanne Klischat" forua="true"/>
    <meta name="description" content="Erreichbarkeitsauskunft und ⏎
    Verzeichnisdienst der Hochschule der Medien, Stuttgart" ⏎
    forua="true"/>
    <meta name="keywords" content="Erreichbarkeit, Availability, ⏎
    Verzeichnisdienst" forua="true"/>
    <meta name="language" content="de" forua="true"/>
    <meta name="organization" content="Fachhochschule Stuttgart - ⏎
    Hochschule der Medien, Fachbereich 3" forua="true"/>
    <meta name="robots" content="index, nofollow" forua="true"/>
  </head>
```

Der Kopfbereich des Decks enthält nur Meta-Angaben. Sie sind fast identisch mit den Meta-Angaben für HTML-Dokumente und enthalten Angaben zu der Autorin des Dokumentes, zu Stichwörtern, Sprache und zugehöriger Organisation, sowie eine Beschreibung und eine Anweisung für Suchmaschinen, das Dokument zu indexieren, jedoch nicht den Links zu folgen.

Das Attribut `forua="true"` bedeutet, daß die Meta-Angaben an den User Agent geschickt werden. Dies ist nötig, wenn Suchmaschinen auf die Meta-Angaben zugreifen sollen. Bei der Übertragung des WML-Decks auf Mobiltelefone filtert meist das Gateway die Meta-Angaben heraus.

6.3.1.2 Begrüßung und Erklärung

```
<card id="start" title="Willkommen" ontimer="#erkl">
❶ <timer name="t" value="30"/>
  <p>
    <b>Willkommen bei den WAP-Services der HdM Stuttgart!</b>
    <br/>
    <br/>
❷   <a href="#erkl">Weiter</a>
    <br/>
  </p>
</card>
```

Die Begrüßungs-Card hat eine automatische Weiterleitung. Aktiviert der Benutzer nicht innerhalb der im `<timer>`-Tag ❶ angegebenen Zehntelsekunden den Link ❷, wird er auf die im Eventhandler `ontimer` angegebenen Card weitergeleitet.

```
<card id="erkl" title="Was">
  <p>
    Sie geben den Namen ein, w&#228;hlen den Zeitpunkt aus, und diese
    Auskunft sagt Ihnen, ob die Person an der Hochschule erreichbar
    ist.
    <br/>
    <a href="#name">Weiter</a><br/>
    <a href="credits.wml">&#220;ber...</a>
  </p>
</card>
```

Die Erklärungs-Card ist recht unspektakulär, sie enthält zwei Links, die zu einer kurzen Zusatzinformation zur Erreichbarkeitsauskunft (*credits.wml*) und zur nächsten Card führen. Im Text sieht man, daß in WML Sonderzeichen (also auch deutsche Umlaute) wie in HTML als Entitäten codiert werden müssen.

6.3.1.3 Card für die Eingabe des Namens

```
<card id="name" title="Wen" newcontext="true">
  <p>Anfang des Nachnamens:<br/>
❶     -&gt; <input type="text" name="idoz" emptyok="false"
      format="A*a" maxlength="5" size="5"/><br/>

❷   <anchor>
      Weiter
        <go href="#datausw">
❸         <setvar name="ddoz" value="$(idoz)"/>
        </go>
     </anchor>

  </p>
</card>
```

Hier wird ein Eingabefeld ❶ definiert, dazu einen Link ❷, der auf die nächste Card verweist und gleichzeitig den vom User eingegebenen Wert des Eingabefeldes in der Variable `ddoz` speichert ❸.

Eingabefelder werden in WML über den `<input>`-Tag erstellt. Die Attribute `maxlength` und `size` geben maximale Länge des einzugebenden Wertes und Größe des Feldes bei der Anzeige an (werden aber von fast keinem Endgerät unterstützt). `emptyok` gibt an, ob eine Eingabe erforderlich ist (false = ja, true = nein). Interessant ist das Attribut `format`. Über `format` kann das Format der einzugebenden Zeichen bestimmt werden. Tabelle 16 zeigt, wie die Zeichenkette für den Wert des Attributs zusammengesetzt wird.

Tabelle 16: Mögliche Codes für den Wert des Attribut `format` im `<input>`-Tag

Code	**Bedeutung**
a	Kleinbuchstabe oder Interpunktionszeichen, keine Ziffer
A	Großbuchstabe oder Interpunktionszeichen, keine Ziffer
N	Ziffer
x	Kleinbuchstabe, Interpunktionszeichen oder Ziffer
X	Großbuchstabe, Interpunktionszeichen oder Ziffer
m	beliebiges Zeichen, Großbuchstaben werden bei der Eingabe in Kleinbuchstaben umgewandelt
M	beliebiges Zeichen, Kleinbuchtaben werden bei der Eingabe in Großbuchstaben umgewandelt
*	der dahinterstehende Code darf beliebig oft wiederholt werden (kann nur einmal am Ende der Zeichenkette angewendet werden)
1-9	der dahinterstehende Code wird so oft wiederholt, wie die Zahl angibt (kann nur einmal am Ende der Zeichenkette angewendet werden)
\	das auf den Backslash folgende Zeichen wird bei dem Versand des Eingabefeldes eingefügt: gibt der Benutzer „0815“ in ein Feld ein, wird dies bei der format-Angabe „NN\/NN“ umgewandelt in „08/15“

Der Wert „A*a“ bedeutet also: ein Großbuchstabe/Interpunktionszeichen und beliebig viele Kleinbuchstaben/Interpunktionszeichen. Ein eingetippter Name beginnt automatisch mit einem Großbuchstaben. Leider wird das `format`-Attribut von einigen Browsern und Endgeräten nicht unterstützt (beispielsweise dem Wireless Companion).

Hat der Benutzer Buchstaben eingegeben, werden diese in der WML-Variable `$idoz` gespeichert, sobald der Benutzer den Link ❷ für die nächste Karte aktiviert.
`<anchor>...<go href=“...“></go></anchor>` ist die ausführliche Schreibweise von `<a href=“...“>...</a>`. Sie wird eingesetzt, sobald ein Link nicht nur einen Verweis darstellen soll, sondern noch eine Zusatzfunktion ausführen soll, wie das Setzen einer Variablen oder die Übergabe von Formulardaten.
`<setvar name=“Variablenname“ value=“Wert“>` setzt eine WML-Variable. WML-Variablen werden benutzt, wenn Benutzereingaben dynamisch in das Dokument eingefügt oder Formulardaten versandt werden sollen. Der Inhalt von Eingabefeldern wird unter dem Namen des Feldes für die aktuelle Card zur Verfügung gestellt. Damit er auf einer nachfolgenden Card ausgegeben werden kann, wird er in einer Variablen gespei-

chert. Variablennamen dürfen Buchstaben, Zahlen und den Unterstrich enthalten, dürfen jedoch nicht mit einer Zahl beginnen. Der Zugriff auf Variablen erfolgt über den Namen mit vorangestelltem Dollarzeichen. Runde Klammern um den Namen sind optional, werden aber empfohlen, damit der Quelltext übersichtlicher wird.

6.3.1.4 Card für die Auswahl des Zeitpunktes

```
<card id="datausw" title="Wann">
❶ <do type="prev" label="Zur&#252;ck">
    <go href="#name"/>
  </do>

❷   <p>
    F&#252;r welchen Zeitpunkt wollen Sie Informationen &#252;ber
    "$(ddoz)..."?<br/>
```

Bei ❶ wird ein Softkey mit einem "Zurück"-Verweis belegt (sofern es der User Agent unterstützt). WML hat in einem Tag realisiert, was in HTML-Dokumenten nur über die JavaScriptFunktion `history.back()` bewerkstelligt werden kann.

Im Text der Card wird die vorher gesetzte Variable `ddoz` ausgelesen ❷.

```
    <anchor>
      Nur Kontaktdaten
❸       <go method="post" href="hauptsuche.php">
❹           <postfield name="pdoz" value="$(ddoz)"/>
            <postfield name="typ" value="ad"/>
        </go>
    </anchor>
<br/>
```

Das Absenden von Formulardaten geschieht in WML über die Kombination der Tags `<anchor>`, `<go>` und `<postfield>`. Im Unterschied zu HTML-Formularen ist es ein ganz normaler Link, kein Button.

Im `<go>`-Tag ❸ wird das Skript angegeben, an das die Formulardaten gesendet werden sollen, sowie die Übergabe-Methode. Bei der Verwendung von PHP macht es für die Weiterverarbeitung im Gegensatz zu z. B. Perl keinen Unterschied, ob Daten über POST (im HTTP-Header) oder GET (in der URL) übergeben werden.

Jedes `<postfield>`-Tag enthält ein Formularfeld und dessen Wert. Bei HTML-Formularen werden die Namen und Werte aller Eingabefelder, Checkboxes und Radiobuttons zwischen `<form>` und `</form>` übergeben. In WML gibt es keinen vergleichbaren Tag. Daher muß im Absende-Link angegeben werden, welche Name-Wert-Paare übergeben werden sollen.

```
    <anchor>
❺     Jetzt
```

```
          <go method="post" href="hauptsuche.php">
            <postfield name="typ" value="plus"/>
            <postfield name="pdoz" value="$(ddoz)"/>
            <postfield name="pplus" value="0"/>
          </go>
     </anchor>
<br/>
     <anchor>
❻      Morgen, diese Zeit
          <go method="post" href="hauptsuche.php">
            <postfield name="typ" value="plus"/>
            <postfield name="pdoz" value="$(ddoz)"/>
            <postfield name="pplus" value="24"/>
          </go>
     </anchor>
<br/>
```

Die beiden Schnellwahlen „Jetzt“ ❺ und „Morgen, diese Zeit“ ❻ sind ebenfalls keine normalen Links sondern senden Formulardaten ab (eines Formulars, das für den Benutzer nicht als solches erkennbar ist).

```
❼    <a href="#dateing">Zeitpunkt eingeben</a><br/>
     in x Stunden:<br/>
     -&gt; <input type="text" name="stplus" format="*N" maxlength="2"
     size="2"/>
<br/>
<br/>
     <anchor>
❽      [..OK..]
          <go method="post" href="hauptsuche.php">
            <postfield name="typ" value="plus"/>
            <postfield name="pdoz" value="$(ddoz)"/>
            <postfield name="pzeit" value="$(stplus)"/>
          </go>
     </anchor>
<br/>
</p>
</card>
```

Das einzige für den Benutzer erkennbare Formularelement dieser Card ist ein Eingabefeld um Stunden zum jetzigen Zeitpunkt zu addieren ❼. Das Absenden geschieht wieder über einen Link ❽.

6.3.1.5 Card für die Eingabe des Zeitpunktes

Auf dieser Card kann ein Benutzer einen genauen Zeitpunkt angeben, d. h. er gibt dem Tag an, wählt den Monat aus und trägt die Stunde ein.

```
<card id="dateing" title="Wann">
```

```
❶ <do type="prev" label="Zur&#252;ck">
     <go href="#datausw">
     </go>
  </do>
<p>
  Am <br/>
❷ Tag: <input type="text" name="itag" format="*N" maxlength="2"
  size="2"/><br/>
  Monat: 
❸ <select name="imonat" multiple="false">
     <option value="10">Okt_2001</option>
     <option value="11">Nov_2001</option>
     <option value="12">Dez_2001</option>
     <option value="1">Jan_2002</option>
     <option value="2">Feb_2002</option>
     <option value="3">M&#228;rz_2002</option>
  </select>
  <br/>

  Uhrzeit (Stunde): 
❹    <input type="text" name="iuhr" format = "*N" maxlength="2"
     size="2"/>
❺    <anchor>[..OK..]
       <go href="hauptsuche.php">
         <postfield name="typ" value="eing"/>
         <postfield name="pdoz" value="$(ddoz)"/>
         <postfield name="ptag" value="$(itag)"/>
         <postfield name="pmonat" value="$(imonat)"/>
         <postfield name="puhr" value="$(iuhr)"/>
       </go>
     </anchor>
</p>
</card>
</wml>
```

Um die Navigation für den Benutzer einfacher zu gestalten, wurde auch hier ein Softkey mit einem „Zurück“-Link belegt ❶. Der Tag wird in ein auf Ziffern beschränktes Eingabefeld eingetragen ❷, ebenso die Uhrzeit ❹. Der Monat wird aus einer Liste ausgewählt ❸. Auswahllisten werden ähnlich wie in HTML erstellt: `<select name=“...“> <option value=“...“>Auswahloption </option> </select>`. Es gibt nur zwei Unterschiede: `<option>` benötigt einen Endtag und muß eine Wertangabe enthalten. Wie gehabt wird auch diese Formular über einen Link versandt ❺, Tag, Monat und Stunde werden in drei Feldern übergeben. Es verwundert vielleicht, daß die Auswahl der Monate nicht dynamisch generiert wird. Dies hat einen Grund: Die Stundenplan-Datenbank, die als Datengrundlage dient, betrifft nur das Wintersemester 2001/2002. Für das nächste Semester wird eine neuen Datenbank angelegt, die wahrscheinlich auch

eine andere Tabellenstruktur haben wird (wie die Vorgänger der jetzigen Datenbank ebenfalls andere Strukturen hatten als die jetzige).

6.3.2 Auswertendes Skript *hauptsuche.php*

Alle Eingaben, die ein Benutzer macht und alle Auswahlen, die er trifft werden an das Skript *hauptsuche.php* übergeben. Je nach Art der Anfrage, kann es zwei Auskünfte erteilen:

- Erreichbarkeitsinformationen
- Kontaktdaten

Hierzu greift es auf vier Include-Dateien zurück, die je nach Anfrage eingebunden werden: *suche_abwesenheit.inc* (Überprüfung, ob Dozent abwesend ist), *suche_lv.inc* (Überprüfung, ob Dozent jetzt/früher/später eine Lehrveranstaltung hält), *suche_adresse.inc* (Zusammenstellen der Kontaktdaten) und *hilf_funk.inc* (Hilfsfunktionen, z. B. Umformatierung des Namens, Austausch von Umlauten). Ich gehe später noch auf jede der Dateien im Einzelnen ein.

hauptsuche.php hat vier Aufgaben:

- Plausibilitätskontrolle für die übergebenen Parameter
- Vereinheitlichung der Zeitpunkteingaben im Unix-Timestampformat
- Suche nach gewünschtem Dozenten
- Ausgabe der Erreichbarkeitsinformationen, meist Rückgabewerte der Funktionen aus inkludierten Dateien, in eine oder mehrere Cards (je nach Anzahl der gefundenen Dozenten)
- Ausgabe des Doctype, Kopfbereichs, des Templates für das anzuzeigende WML-Dokument

6.3.2.1 Plausibilitätskontrolle und Normierung des Zeitpunkts

Bei der Plausibilitätskontrolle und der Normierung von Zeitpunktangaben handelt es sich um zwei verschieden Aufgaben von *hauptsuche.php*, die sich jedoch am einfachsten parallel lösen lassen.

Die Plausibilitätskontrolle betrifft die folgenden Benutzereingaben:

- Dozentname (`$pdoz`)
- zu addierende Stunden (`$pplus`)
- Eingabe des Tages (`$ptag`)
- Eingabe der Uhrzeit (`$puhr`)

Desweiteren wird in diesem Teil die aktuelle Zeit als Unix-Timestamp generiert ❸. Zu ihr werden dann die zu addierenden Stunden hinzugezählt.

Ich habe mich für den Unix-Timestamp entschieden, da sich aus ihm mit der PHP-Funktion `date()`[16] die verschiedensten Angaben extrahieren lassen (Datum, Wochentag, Stunde). Der Unix-Timestamp hat den weiteren Vorteil, daß es keinerlei Probleme bereitet, Stunden oder Tage zu addieren (solange sie in Sekunden umgerechnet werden). Einziger Nachteil, den ich erst nach dem Test des PHP-Skriptes auf dem Lehre-Server feststellen konnte, ist, daß auf dem Lehre-Server (im Gegensatz zu meinem Rechner) bei der Erstellung des Timestamps die Sommerzeit (MESZ) nicht berücksichtigt wird. Inzwischen habe ich dieses Problem behoben.

Alle weiteren Zeitpunktangaben werden, mit der PHP-Funktion `mktime()` in einen Unix-Timestamp umgewandelt. Zur Unterscheidung sind Anweisungen, die die Zeitpunkt-Normierung betreffen, kursiv formatiert.

```
❶ if ($pdoz == '')
  die ("<card><p>Bitte geben Sie f&#252;r den Namen des  ⏎
       Dozenten mindestens einen Anfangsbuchstaben ein.<br/>  ⏎
       <a href=\"index.php#name\">Zur&#252;ck</a></p></card></wml>");

❷ if ($typ == "plus")              // Stunden >= 0 sollen zu “Jetzt”
    {                              // addiert werden
❸   $pzeit = time();
    //Sommerzeit korrigieren
    if (date("I", $pzeit) == "1")
    $pzeit += 60*60;

    if ($pplus == '')
    $pplus = 0;

    $timestamp = $pzeit + ($pplus * 3600);
    }
  elseif ($typ == "eing")      // Usereingabe des Zeitpunktes
  {
❹   if ($ptag == '')
    die ("<card><p>Bitte geben Sie ein g&#252;ltiges  ⏎
         Datum ein.<br/><a href=\"index.php#dateing\">  ⏎
         Zur&#252;ck</a></p></card></wml>");

     switch ($pmonat)
     {
     case 10:
         $monat=10;
         $jahr=2001;
         break;
     case 11:
```

[16] Für mehr Informationen zu dieser PHP-Funktion und allen anderen siehe Schmid 2001

```
         $monat=11;
         $jahr=2001;
         break;
      case 12:
         $monat=11;
         $jahr=2001;
         break;
      case 1:
         $monat=1;
         $jahr=2002;
         break;
      case 2:
         $monat=2;
         $jahr=2002;
         break;
      case 3:
         $monat=3;
         $jahr=2002;
         break;
      }

❺   if (!checkdate($monat, $ptag, $jahr))
      {
      if ($ptag == 0)
      $ptag = 1;

      if ($ptag > 28)
          {
          $monat++;
          $ptag = 0;
          }
      }

❻   if ($puhr == '')
    $puhr = 1;

    if ($puhr > 24)
    $puhr = 24;

  $timestamp = mktime($puhr, 0, 0, $monat, $ptag, $jahr);
  }
```

Als erstes wird überprüft, ob überhaupt ein Name eingegeben wurde, ansonsten bricht die Ausführung des Skripts mit der Ausgabe einer Fehlermeldung ab ❶. Diese Funktion hätte auch über WMLScript gelöst werden können. Die folgende Funktion würde über den `<anchor>`-Tag, der jetzt auf die nächste Card führt, aufgerufen.

```
extern function name_test()
```

```
{
var name = WMLBrowser.getVar("idoz");
if (name == "")
   {
   Dialogs.alert("Bitte geben Sie einen Namen ein. ");
   }
else
   {
   WMLBrowser.setVar("pdoz", name);
   WMLBrowser.go("index.php#datausw");
   }
}
```

Der Grund, warum ich auf die WMLScript-Variante verzichtet habe ist, daß es unmöglich ist für einen User Agent, der kein WMLScript unterstützt, zur nächsten Card zu gelangen. Seine Benutzer kann dadurch die Auskunft nicht nutzen.

Bei der Plausibilitätskontrolle der Zeitangaben war mir wichtig, daß die Auskunft sehr fehlertolerant reagiert, d. h. der Benutzer soll nicht sofort mit einer Fehlermeldung konfrontiert werden (die, da serverseitig erzeugt, erst nach einer Wartezeit erscheint), sondern eine falsche Eingabe wird interpretiert und abgeändert, um möglichst doch die Information auszugeben, die der Benutzer eigentlich anfordern wollte ❷-❺.

Einige der überprüften Fälle dürften nicht eintreten, da im WML-Dokument schon Vorkehrungen getroffen wurden (z. B. `emptyok = „false“`), jedoch kann es vorkommen, daß der eine oder andere User Agent diese Attribute nicht unterstützt.

6.3.2.2 Dozenten-Suche

Es wird gesucht, ob Dozenten in der Datenbank verzeichnet sind, deren Nachname mit den eingegebenen Buchstaben anfängt:

```
$verb = mysql_connect('localhost', 'user','passwort');
$erg_doz = mysql_db_query("ws2001",  ⏎
                          "SELECT nummer, Name, Vorname  ⏎
                          FROM dozadr  ⏎
                          WHERE Name LIKE '$post_doz%'  ⏎
                          ORDER BY Name", $verb);
```

Einer mySQL-Datenbankabfrage geht in PHP zuerst der Verbindungsaufbau zum Datenbankmanagementsystem voraus: `mysql_connect(host, user, passwort)`. Dann wird über `mysql_select_db(Datenbank)` die Datenbank ausgewählt und mit `mysql_query(SQL)` die SQL-Abfrage gestellt. Beide Funktionen können auch in `mysql_db_query(Datenbank, SQL)` kombiniert werden.

Das Ergebnis einer Anfrage muß in einer Schleife abgeholt werden. Der Programmierer hat die Auswahl, es als indiziertes Array (`mysql_fetch_row(Ergebniskennung)`), assoziatives Array (`mysql_fetch_array(Ergebniskennung)`) oder Objekt (`mysql_fetch_object(Ergebniskennung)`) zu holen.

Der Vollständigkeit halber müsste nach erfolgter Abfrage und Ergebnis-Abholung das Ergebnis wieder freigegeben werden (`mysql_free_result(Ergebniskennung)`) und die Verbindung geschlossen (`mysql_close(Verbindungskennung)`). Für gewöhnlich ist die jedoch nicht nötig, da nach Ausführung des Skripts das Ergebnis automatisch freigegeben wird und die Verbindung geschlossen.

In diesem Fall wird allerdings zuerst die Anzahl der gefundenen Ergebnisse gespeichert. von ihr ist abhängig, ob und wie die Erreichbarkeitsinformationen ausgegeben werden.

```
$erg_doz_num = mysql_num_rows($erg_doz);
if (($erg_doz_num<=8) and ($erg_doz_num>0))
  {...}
elseif ($erg_doz_num>8)
  {...}
elseif ($erg_doz_num==0)
  {...}
```

Wird kein Dozent gefunden, erhält der Benutzer eine entsprechende Meldung:

```
elseif ($erg_doz_num==0)
  {
  echo "<card id=\"fehler\">
  <p>Leider wurde kein Dozent mit dem Namen '$pdoz...'  ⏎
  gefunden.  ⏎
  </p>  ⏎
  </card>";
  }
```

Bei mehr als zehn gefundenen Dozenten erhält der Benutzer eine Liste aller gefundener Dozenten. Durch Aktivieren eines angezeigten Links wird eine verfeinerte Suchanfrage wieder an *hauptsuche.php* übergeben.

Acht Dozenten sind keine willkürlich ausgewählte Begrenzung. Tests haben gezeigt, daß, wenn die Erreichbarkeitsinformationen von acht Dozenten in einem WML-Deck ausgegeben werden, das Deck die Größenbegrenzung genau nicht überschreitet.

```
elseif ($erg_doz_num > 8)
  {
  echo "<card id=\"zuviele\">  ⏎
        <p>Bitte w&#228;hlen Sie aus:<br/>\n";

  for ($l=0; $l<mysql_num_rows($erg_doz); $l++)
    {
    $doz_einzel = mysql_fetch_object($erg_doz);
    $doz_name = titel($doz_einzel->Vorname, $doz_einzel->Name);

    if ($typ == "plus" or $typ == "eing")
    $q_string = "pdoz=$doz_einzel->Name&amp;  ⏎
                 typ=plus&amp;pzeit=$timestamp&amp;  ⏎
```

```
                pplus=0";

    if ($typ == "ad")
    $q_string = "pdoz=$doz_einzel->Name&amp;typ=ad";

    echo "<a href=\"$PHP_SELF?$q_string\">";
    echo $doz_name[gesamt]."</a><br/>\n";
    }
  echo "</p>  ↵
  </card>";
  }
```

Bei einer Suche nach den Kontaktdaten aller Dozenten mit „S“, ergäbe sich dann folgende Ausgabe:

```
<card id="zuviele">
<p>Bitte w&#228;hlen Sie aus:<br/>
<a href="?pdoz=Sailer&amp;typ=ad">Klaus Sailer</a><br/>
<a href="?pdoz=Scharlau&amp;typ=ad">Dr. Ulf Scharlau</a><br/>
...
</p>
</card>
```

6.3.2.3 Ausgabe der Erreichbarkeitsinformationen

Werden ein bis zehn Dozenten gefunden, werden die zugehörigen Erreichbarkeitsinformationen oder Kontaktdaten ausgegeben. Wünscht der Benutzer Erreichbarkeitsinformationen, wird für jeden gefundenen Dozenten der gesamte Name zusammengesetzt ❶, dann die Abwesenheit geprüft ❷ (`abwesenheit()` stammt aus *suche_abwesenheit.inc*). Ist der Dozent abwesend, bekommt der Benutzer eine Meldung, ansonsten wird mit der Funktion `lv()` aus *suche_lv.inc* die Erreichbarkeitsinformation erstellt ❸. Bei mehreren Dozenten wird zuerst eine Übersichts-Card erstellt ❹ und dann die Erreichbarkeitsinformation auf je einer Card angehängt ❺. Am Ende der Erreichbarkeitsinformation steht ein Link zu den Kontaktdaten des Dozenten ❻.

```
  if (($typ == "plus") or ($typ == "eing"))
    {
    for ($i=0; $i<$erg_doz_num; $i++)
      {
      $doz_einzel = mysql_fetch_object($erg_doz);
❶     $name = titel($doz_einzel->Vorname, $doz_einzel->Name);
      $dozent[$i] = $name[gesamt];
      $dozent_name[$i] = $name[nachname];
      $doz_num[$i] = $doz_einzel->nummer;

❷     $abwesenheit = abwesenheit($doz_einzel->nummer, $timestamp);
       if($abwesenheit)
       $ausgabe[$i] = "$dozent[$i] ist zum gew&#252;nschten  ↵
```

```
                Zeitpunkt leider nicht anwesend $abwesenheit";

          else
❸         $ausgabe[$i] = lv($doz_einzel->nummer, $timestamp);
          }
        if (count($ausgabe) > 1)
           {
❹          echo "<card id=\"doz_auswahl\">  ⏎
                <p>Bitte w&#228;hlen Sie aus:<br/>\n";

           for ($j=0; $j<count($ausgabe); $j++)
              {
              echo "<a href=\"#doz$doz_num[$j]\">$dozent[$j]</a><br/>\n";
              }
           echo "</p></card>\n";
           }

        for ($k=0; $k<count($ausgabe); $k++)
           {
❺          echo "<card id=\"doz$doz_num[$k]\">
                <p>
                $ausgabe[$k]\n<br/>  ⏎
❻               <a href=\"$PHP_SELF?  ⏎
                typ=adresselink&amp;pdoz=$dozent_name[$k]\">  ⏎
                Kontaktdaten</a>  ⏎
                <br/><br/>  ⏎
                <small>Alle Angaben ohne Gew&#228;hr.</small>  ⏎
                </p></card>\n";
         }
   }
```

Angenommen, ein Benutzer suchte nach Erreichbarkeitsinformationen für „Ra“ am 03. November 2001, wird folgendes ausgegeben:

```
<card id="doz_auswahl">
<p>Bitte w&#228;hlen Sie aus:<br/>
<a href="#doz85">Dr. Christian Rathke</a><br/>
<a href="#doz86">Dr. Wolfgang Ratzek</a><br/>
</p>
</card>

<card id="doz85">
<p>
<i>Dr. Christian Rathke</i> hat an diesem Tag keine
Lehrveranstaltungen. (2001-11-03)
<br/>
<a href="?typ=adresselink&amp;pdoz=Rathke">
Kontaktdaten
```

```
</a>
<br/><br/><small>Alle Angaben ohne Gew&#228;hr.</small>
</p></card>
... // Vergleichbare Card für Dr. Ratzek folgt
```

Will ein Benutzer den Verzeichnisdienst in Anspruch nehmen, also Kontaktdaten abrufen, werden die entsprechenden Cards über die Funktion `adresse()` aus *suche_adresse.inc* erzeugt. Der Unterschied zwischen `$typ == „adresse“` und `$typ == „adresselink“` ist, daß bei `„adresselink“` der `adresse()`-Funktion eine URL übergeben wird, die bei der Anzeige der Kontaktdaten auf einen Softkey gelegt wird. Ruft ein Benutzer die Kontaktdaten aus den Erreichbarkeitsinformationen eines Dozenten auf, kann er über diesen Softkey beispielsweise wieder zurück zu den Informationen gelangen.

```
elseif ($typ == "ad" or $typ == "adresselink")
  {
  include_once('suche_adresse.inc');
  $doz_einzel = mysql_fetch_object($erg_doz);

  if ($typ == "ad")
  echo adresse($doz_einzel->nummer, "wml_suche.php");

  if ($typ == "adresselink")
  echo adresse($doz_einzel->nummer, "$PHP_SELF?  ↵
               pdoz=$doz_einzel->Name&amp;typ=plus&amp;  ↵
               pzeit=$timestamp&amp;pplus=0");
  }
```

6.3.3 Inkludierte Dateien

6.3.3.1 hilf_funk.inc

hilf_funk.inc enthält zwei Hilfsfunktionen, die ausgelagert wurden, um den Quellcode der anderen Dateien übersichtlicher zu halten. Einmal ist dies `titel()`, um einen Namen zu formatieren, und `umlaute()` um deutsche Umlaute in die entsprechenden WML-Entitäten zu konvertieren.

`titel()` dient dazu, eine Unzulänglichkeit der Stundenplan-Datenbank auszugleichen. In der Datenbank wird der Titel einer Person an den Vornamen angehängt: „Wolf-Fritz, Dr.“. Ich will jedoch, daß Titel, Vor- und Nachname einzeln ausgegeben werden können, um z. B. „Dr. Wolf-Fritz Riekert“ zusammenzusetzen.

Zuerst werden Titel und Vornamen getrennt. Hierzu ist eine Fallunterscheidung nötig ❶, da es bei einigen fehlerhaften Einträgen vorkommt, daß das trennende Komma fehlt. Nach der Aufbereitung von Titel, Vor- und Nachname ❷ wird der gesamte Name zusammengesetzt: ein Mastertitel wird am Ende angehängt ❸, alle anderen Titel vorne ❹.

```
function titel($vornametitel, $nachname)
```

```
{
❶ if (ereg(",", $vornametitel))
  list($vorname, $titel) = explode(",", $vornametitel);

  elseif (ereg(" ", $vornametitel))
  list($vorname, $titel) = explode(" ", $vornametitel);

  else
  $vorname = $vornametitel;

❷ $name[vorname] = trim($vorname);
  $name[vorname] = umlaute($name[vorname]);
  $name[titel] = trim($titel);
  $name[titel] = umlaute($name[titel]);
  $name[nachname] = umlaute($nachname);

❸ if (ereg("^M.", $name[titel]))
  $name[gesamt] = $name[vorname]." ".$name[nachname].",
  ".$name[titel];

  if (empty($name[titel]))
  $name[gesamt] = $name[vorname]." ".$name[nachname];

  else
❹ $name[gesamt] = $name[titel]." ".$name[vorname]." ".$name[nachname];

  $name[gesamt] = trim($name[gesamt]);
  return $name;
}
```

`umlaute()` ist eine einfache Suchen-und-Ersetzen-Funktion:

```
function umlaute($umlautstring)
{
$umlautstring = ereg_replace("ä", "&#228;", $umlautstring);
$umlautstring = ereg_replace("Ä", "&#196;", $umlautstring);
$umlautstring = ereg_replace("ö", "&#246;", $umlautstring);
$umlautstring = ereg_replace("Ö", "&#214;", $umlautstring);
$umlautstring = ereg_replace("ü", "&#252;", $umlautstring);
$umlautstring = ereg_replace("Ü", "&#220;", $umlautstring);
$umlautstring = ereg_replace("ß", "&#223;", $umlautstring);
$umlautstring = ereg_replace("[èéê]", "e", $umlautstring);
$umlautstring = ereg_replace("[ÈÉÊ]", "E", $umlautstring);

return $umlautstring;
}
```

6.3.3.2 suche_abwesenheit.inc

suche_abwesenheit.inc enthält nur die Funktion `abwesenheit()`. Diese Funktion fragt in der ergänzenden Datenbank ab, ob für einen Dozent zu einem bestimmten Zeitpunkt eine Abwesenheit eingetragen ist.

Zuerst werden der am weitesten vor dem übergebenen Zeitpunkt liegende Beginn ❶ und das am weitesten danach liegende Ende einer Abwesenheit gesucht ❷. Die Ergebnisse dieser Abfragen werden umformatiert und zurückgegeben ❸. Wird kein Ergebnis gefunden, gibt die Funktion keinen Wert zurück.

```
function abwesenheit($doz_einzel_nummer, $timestamp)
{
$user="user";
$passwd="passwort";
$host="localhost";
$db="stplankontakt";  // ergänzende Datenbank

❶ $sql_beginn="SELECT MIN(beginn) FROM abwesenheit  ⏎
           WHERE (dozenten_nummer=$doz_einzel_nummer)  ⏎
           AND (beginn < $timestamp)";

❷ $sql_ende="SELECT MAX(ende) FROM abwesenheit  ⏎
           WHERE (dozenten_nummer=$doz_einzel_nummer)  ⏎
           AND (ende > $timestamp)";

$verb_abwesenheit = mysql_connect($host, $user, $passwd);
$db_abwesenheit = mysql_select_db($db, $verb_abwesenheit);

$ergebnis_beginn = mysql_query($sql_beginn, $verb_abwesenheit);
$ergebnis_ende = mysql_query($sql_ende, $verb_abwesenheit);

list($beginn) = mysql_fetch_row($ergebnis_beginn);
list($ende) = mysql_fetch_row($ergebnis_ende);
mysql_free_result($ergebnis_beginn);
mysql_free_result($ergebnis_ende);

if (! empty($beginn) and ! empty($ende))
❸ $return = "(abwesend von ".date("d.m.Y" , $beginn).  ⏎
         " bis ".date("d.m.Y" , $ende).").";

elseif (! empty($beginn) and empty($ende))
❸ $return = "(abwesend seit ".date("d.m.Y", $beginn).").";

elseif (empty($beginn) and ! empty($ende))
❸ $return = "(abwesend bis ".date("d.m.Y", $ende).").";
```

```
mysql_close($verb_abwesenheit);
❸ return $return;
}
```

6.3.3.3 suche_adresse.inc

suche_adresse.inc enthält die beiden Funktionen `adresse()` und `wtai()`, sowie die Klasse `visitenkarte`. `visitenkarte` speichert alle Angaben für die Kontaktdaten bzw. die vCard-Visitenkarte eines Dozenten ❶, einige werden zugewiesen, andere sind bereits vorbelegt über den Konstruktor `visitenkarte()` ❷.

```
class visitenkarte
{
❶ var $vorname;
  ...              // gekürzt aus Platzgründen
  var $vcard;


❷ function visitenkarte()
  {
  $this->vcard = "";
  $this->strasse_dienst = "Wolframstr. 32-34";
  $this->ort_dienst = "Stuttgart";
  $this->plz_dienst = "70191";
  $this->diensttel = "0711-25706-0";
  $this->url = "http://www.hdm-stuttgart.de";
  }
```

Die Klasse `visitenkarte` enthält drei Methoden. `zusammensetzen()` generiert eine Visitenkarte nach dem vCard-Standard, `speichern()` speichert diese im Dateisystem ab, damit sie ein Benutzer herunterladen kann und `ausgabe()` gibt die Kontaktdaten als WML-Card aus.

```
function zusammensetzen()
{
$this->vcard = "begin:vcard\n";
$this->vcard.= "n:$this->nachname;$this->vorname;;$this->titel\n;
$this->vcard.= "fn:$this->vorname $this->nachname\n";
...              // gekürzt aus Platzgründen
$this->vcard.= "version:2.1\n";
$this->vcard.= "x-mozilla-html:FALSE\n";
$this->vcard.= "end:vcard";
}

function speichern()
{
$dateiname = "vcf/".$this->vorname.$this->nachname.".vcf";
$dateiname = ereg_replace("[-]", "", $dateiname);
$verzeichnis = opendir("vcf");
```

```
$vcf = fopen($dateiname, "w");
fwrite($vcf, $this->vcard);
fclose($vcf);
chmod($dateiname, 0755);
closedir($verzeichnis);
}
```

Hier ist ein Beispiel für eine fertige vCard:

```
begin:vcard
n:Riekert;Wolf-Fritz;;Dr.
fn:Wolf-Fritz Riekert
org:Fachhochschule Stuttgart - Hochschule der Medien
title: Dr.
tel;cell;voice:
tel;work;fax:
tel;work;voice:25706-185
tel;home;voice:0731/36456
url;;work:http://v.hbi-stuttgart.de/~riekert/
adr;;work:;N408(W);Wolframstr. 32-34;Stuttgart;;70191;;;
adr;;home:;;Straßburgweg 2;Ulm;;89077;;;
email;internet:riekert@hbi-stuttgart.de
note:Diese VCard stammt aus dem WAP-Verzeichnisdienst der
Erreichbarkeitsauskunft der Fachhochschule Stuttgart - Hochschule der
Medien
version:2.1
x-mozilla-html:FALSE
end:vcard
```

In diesem Format bekommt sie der Benutzer allerdings nicht zu Gesicht. Er sieht nur die von `ausgabe()` erzeugte WML-Card. Da nicht alle Daten, die in dieser WML-Visitenkarte angezeigt werden können, bei jedem Dozenten angegeben sind (insbesodere die Privatanschrift und Privattelefonnummer), bedarf es einiger Fallunterscheidungen bei der Ausgabe ❸-❻.

Jede Telefonnummer, die in der WML-Visitenkarte angezeigt wird, kann über eine WTAI-Bibliotheksfunktion gewählt oder abgespeichert werden. Hierzu greift `ausgabe()` auf die Funktion `wtai()` zurück. `wtai()` erzeugt für jede Telefonnummer eine Card mit den Optionen Wählen/Abspeichern. Diese Cards werden in `$extra_cards` gespeichert ❷. Zusätzlich wird eine Zählervariable ❶ benötigt, die die angehängten Cards zählt. Sie wird `wtai()` als Referenz übergeben, d. h. Änderungen, die `wtai()` an ihr durchführt, sind auch außerhalb der `wtai()`-Funktion gültig.

```
function ausgabe($url)
{

❶ $m=0;                 //Zählervariable für zusätzliche
                        // WML-Cards für Telefonnummern
```

```
❷ $extra_cards = "";   // Zusätzliche Cards für
                       // Telefonnummern-WTAI-Funktionen

  $ausgabe = "<card id=\"adresse$this->nachname\">  ⏎
              <do type=\"prev\" label=\"Zur&#252;ck\">  ⏎
              <go href=\"$url\">  ⏎
              </go>  ⏎
              </do>  ⏎
              <p>  ⏎
              <b>$this->titel $this->vorname $this->nachname</b>  ⏎
              <br/>  ⏎
              <i>Telefon (dienstlich):</i>".  ⏎
              wtai($this->diensttel, $m, $extra_cards,  ⏎
              $this->vorname, $this->nachname)."<br/>\n";

❸ if (! empty($this->privattel))
  $ausgabe .= "<i>Telefon (privat):</i>".  ⏎
              wtai($this->privattel, $m, $extra_cards,  ⏎
              $this->vorname, $this->nachname)."<br/>\n";

❹ if (! empty($this->email))
  $ausgabe .= "<i>E-Mail:</i> <a href=\"mailto:$this->email\">  ⏎
              $this->email</a><br/>\n";

  $ausgabe .= "<i>Website:</i> <a href=\"$this->url\">  ⏎
              $this->url</a><br/>\n";

❺ if (! empty($this->buero))
  $ausgabe .= "<i>B&#252;ro:</i> $this->buero<br/>\n";

  $ausgabe .= "<i>Anschrift (dienstlich):</i><br/>\n";
  $ausgabe .= umlaute($this->strasse_dienst)."<br/>\n";
  $ausgabe .= "$this->plz_dienst ".  ⏎
               umlaute($this->ort_dienst)."<br/>\n";

❻ if (! empty($this->strasse_privat) and ! empty($this->ort_privat))
    {
    $ausgabe .= "<i>Anschrift (privat): </i><br/>\n";
    $ausgabe .= umlaute($this->strasse_privat)."<br/>\n";
    $ausgabe .= "$this->plz_privat ".  ⏎
                 umlaute($this->ort_privat)."<br/>\n";
    }

   $ausgabe .= "</p>  ⏎
               <p align=\"center\">  ⏎
               <a href=\"vcf/";
```

```
   $ausgabe .= ereg_replace("[-]", "",  ⏎
                            $this->vorname.$this->nachname);
   $ausgabe .= ".vcf\">Visitenkarte (VCard)</a><br/>  ⏎
               <small>Download nicht mit jedem Endger&#228;t  ⏎
               m&#246;glich.</small>  ⏎
               </p>  ⏎
               </card>\n";

   $ausgabe .= $extra_cards;
   return $ausgabe;

}
}
```

Die zur obigen vCard korrespondierende WML-Visitenkarte sieht aus wie folgt:

```
<card id="adresseRiekert">
   <do type="prev" label="Zur&#252;ck">
      <go href="wml_suche.php">
      </go>
   </do>
<p>
<b>Dr. Wolf-Fritz Riekert</b><br/>
<i>Telefon (dienstlich):</i><a href="#telnum0">0711/25706-185</a><br/>
<i>Telefon (privat):</i><a href="#telnum1">0731/36456</a><br/>
<i>E-Mail:</i> <a href="mailto:riekert@hbi-stuttgart.de">  ⏎
                        riekert@hbi-stuttgart.de</a><br/>
<i>Website:</i> <a href="http://v.hbi-stuttgart.de/~riekert/">  ⏎
                        http://v.hbi-stuttgart.de/~riekert/</a><br/>
<i>B&#252;ro:</i> N408(W)<br/>
<i>Anschrift (dienstlich):</i><br/>
Wolframstr. 32-34<br/>
70191 Stuttgart<br/>
<i>Anschrift (privat): </i><br/>
Stra&#223;burgweg 2<br/>
89077 Ulm<br/>
</p>
<p align="center">
<a href="vcf/WolfFritzRiekert.vcf">Visitenkarte (VCard)</a><br/>
<small>Download nicht mit jedem Endger&#228;t m&#246;glich.</small>
</p>
</card>
<card id="telnum0">
   <do type="prev" label="Zur&#252;ck">
     <go href="#adresseRiekert">
     </go>
   </do>
```

```
<p>
<b>Wolf-Fritz Riekert</b><br/>
0711/25706-185<br/>
<a href="wtai://wp/mc;%2B4971125706185">W&#228;hlen</a><br/>
<a href="wtai://wp/ap;%2B4971125706185;Riekert">Abspeichern</a><br/>
<small>Bitte beachten Sie, dass nicht jedes Endger&#228;t diese ↵
       Funktionen unterst&#252;tzt.</small>
</p>
</card>
<card id="telnum1">
   <do type="prev" label="Zur&#252;ck">
      <go href="#adresseRiekert">
      </go>
   </do>
<p>
<b>Wolf-Fritz Riekert</b><br/>
0731/36456<br/>
<a href="wtai://wp/mc;%2B4973136456">W&#228;hlen</a><br/>
<a href="wtai://wp/ap;%2B4973136456;Riekert">Abspeichern</a><br/>
<small>Bitte beachten Sie, dass nicht jedes Endger&#228;t diese ↵
       Funktionen unterst&#252;tzt.</small>
</p>
</card>
```

Die bereits erwähnte Funktion `wtai()` ist wie `titel()` zu einem Großteil notwendig, um in der Stundenplan-Datenbank eingetragene Telefonnummern zuerst einmal von Eingabefehlern zu befreien ❶, beispielsweise, wenn „Tel.:“ der Nummer vorangestellt wurde, oder am Ende der Nummern ein HTML-Tag hängt. Bei HdM-Telefonnummern muß außerdem die Stuttgarter Vorwahl ergänzt werden ❷.

Weiterhin müssen die Telefonnummern umformatiert werden, da WTAI-Funktionen sie nur verarbeiten können, wenn sie als MSISDN-Adressen vorliegen ❹. MSISDN steht für Mobile Station International Subscriber Directory Number und bezeichnet die Telefonnummer mit Länderkennung, Netzkennung (Vorwahl) und Rufnummer: +491711234567. Die so formatierte Nummer wird nur in der WTAI-Funktion benutzt, der Benutzer bekommt die Telefonnummer noch mit Trennzeichen zwischen Vorwahl und Rufnummer angezeigt ❸.

Die benutzer- und WTAI-spezifisch formatierten Telefonnummern werden zu einer Card zusammengesetzt ❺ und so zurückgegeben ❻.

```
function wtai($tel_nummer, &$zaehler, &$tel_cards, $vorname, ↵
             $nachname)
{
❶ $tel_nummer = ereg_replace("^[^0-9]*|[^0-9]*$", "", $tel_nummer);
❷ $tel_nummer = ereg_replace("^25706", "0711/25706", $tel_nummer);

❸ $tel_ausg = "<a href=\"#telnum$zaehler\">$tel_nummer</a>";
```

```
❹ $tel_nummer_wtai = ereg_replace("[^0-9]", "", $tel_nummer);
  $tel_nummer_wtai = ereg_replace("^00", "%2B", $tel_nummer_wtai);
  $tel_nummer_wtai = ereg_replace("^0", "%2B49", $tel_nummer_wtai);

❺ $tel_cards .= "<card id=\"telnum$zaehler\">  ⏎
    <do type=\"prev\" label=\"Zur&#252;ck\">  ⏎
      <go href=\"#adresse$nachname\">  ⏎
      </go>  ⏎
    </do>  ⏎
  <p>  ⏎
  <b>$vorname $nachname</b><br/>  ⏎
  $tel_nummer<br/>  ⏎
  <a href=\"wtai://wp/mc;$tel_nummer_wtai\">  ⏎
          W&#228;hlen</a><br/>  ⏎
  <a href=\"wtai://wp/ap;$tel_nummer_wtai;$nachname\">  ⏎
          Abspeichern</a><br/>  ⏎
  <small>Bitte beachten Sie, dass nicht jedes Endger&#228;t diese  ⏎
         Funktionen unterst&#252;tzt.</small>  ⏎
  </p>  ⏎
  </card>\n";

  $zaehler++;
❻ return $tel_ausg;
}
```

Alle für die beschriebenen Funktionen und Methoden nötigen Daten werden von der Funktion `adresse()` aus der Stundenplan-Datenbank und der ergänzenden Tabelle „personen“ abgefragt und als Eigenschaften in einer Instanz der Klasse `visitenkarte` gespeichert. Desweiteren steuert `adresse()` das Zusammensetzen, Abspeichern und Ausgeben einer Visitenkarte.

Zuerst werden alle in der Stundenplan-Datenbank verfügbaren Kontaktdaten abgefragt ❶. Handelt es sich um einen hauptamtlichen Dozenten, wird aus der ertgänzenden Tabelle „personen“ die Website und das Büro abgefragt ❷. Diese Tabelle verwendet andere Nummern als die Stundenplan-Datenbank um Dozenten eindeutig zu kennzeichnen. Daher wird der Name des Dozenten verglichen. Werden dabei mehr als ein Suchergebnis gefunden, wird zusätzlich die E-mail-Adresse hinzugezogen ❸ Ich benutze sie bei der ersten Abfrage nicht, da sie nicht bei allen Dozenten vorhanden ist, und außerdem beim Eintragen des Nachnamen mit einer geringeren Wahrscheinlichkeit ein Fehler gemacht wurde.

Daraufhin wird eine Instanz der `visitenkarte`-Klasse erstellt und Eigenschaften werden ihr zugewiesen ❹. Dabei muß beachtet werden, daß bei einigen Dozenten die Diensttelefonnummer im Tabellenfeld „Diensttel“ steht ❻, bei anderen in „Telefon“ (z. B. wenn keine Privatnummer angegeben ist) mit einer entsprechenden Kennzeichnung

in „TelArt“ ❺. Außerdem kann vorkommen, daß die Mobiltelefonnummer sowohl in „TelMobil“ als auch in „Telefon“ eingetragen wurde ❼. Ähnlich der Privattelefonnummer ist auch eine eventuell vorhandene Privatanschrift gekennzeichnet ❽. Die Dienstanschrift ist in der Datenbank normalerweise vermerkt, wenn sie nicht die Hochschule der Medien ist ❾.

Zuletzt wird dann die Visitenkarte zusammengesetzt, gespeichert und als WML-Card ausgegeben ❿.

```
function adresse($nummer, $aufrufurl)
{
include_once('hilf_funk.inc');

$ad_config[user]="user";
$ad_config[passwd]="passwort";
$ad_config[host]="localhost";

$ad_verb = mysql_connect($ad_config[host], $ad_config[user],  ⏎
                         $ad_config[passwd]);

❶ $ad_erg = mysql_db_query("ws2001", "SELECT Vorname, Name,  ⏎
            Strasse,PLZ, Ort, Telefon, email, Fax, TelDienst,  ⏎
            TelMobil, TelArt, AdrArt, DozArt  ⏎
            FROM dozadr  ⏎
            WHERE nummer = $nummer", $ad_verb);

$ad_ergausg = mysql_fetch_object($ad_erg);

❷ if ($ad_ergausg->DozArt == 'Hauptamtlich')
  {
    $buero_web_erg = mysql_db_query("stplankontakt",  ⏎
                     "SELECT RaumID, Website FROM personen  ⏎
                     WHERE Nachname  ⏎
                     LIKE '$ad_ergausg->Name'", $ad_verb);

    if (mysql_num_rows($buero_web_erg) == 1)
    $buero_ergausg = mysql_fetch_object($buero_web_erg);
    ❸ if (mysql_num_rows($buero_web_erg) > 1)
        {
        $mail_suchmuster = explode("@", $ad_ergausg->email);
        $buero_web_erg = mysql_db_query("stplankontakt",  ⏎
                         "SELECT RaumID, Website, email  ⏎
                         FROM personen  ⏎
                         WHERE  ⏎
                         Nachname LIKE '$ad_ergausg->Name'
                         AND (email LIKE '$mail_suchmuster[0]@%')",
                         $ad_verb);
```

```
        }
  }

❹ $vcard = new visitenkarte;

  $name = titel($ad_ergausg->Vorname, $ad_ergausg->Name);
  $vcard->vorname = $name[vorname];
  $vcard->nachname = $name[nachname];
  $vcard->titel = $name[titel];

❺ if (($ad_ergausg->TelArt == 'dienstlich') and  ↵
   ($ad_ergausg->Telefon != ''))
  $vcard->diensttel = $ad_ergausg->Telefon;

❻ if ($ad_ergausg->TelDienst != '')
  $vcard->diensttel = $ad_ergausg->TelDienst;

❼ if (($ad_ergausg->TelMobil != '') and  ↵
   (($ad_ergausg->TelMobil) != ($ad_ergausg->Telefon)))
  $vcard->mobiltel = $ad_ergausg->TelMobil;

  if (($ad_ergausg->TelArt == 'privat') and ($ad_ergausg->Telefon
  !=''))
  $vcard->privattel = $ad_ergausg->Telefon;

  if ($ad_ergausg->Fax !='')
  $vcard->fax = $ad_ergausg->Fax;

  if ($ad_ergausg->email !='')
  $vcard->email = $ad_ergausg->email;

  if (isset($buero_ergausg) and ($buero_ergausg->Website !=''))
  $vcard->url = $buero_ergausg->Website;

  if (isset($buero_ergausg) and ($buero_ergausg->RaumID != ''))
  $vcard->buero = $buero_ergausg->RaumID;

❽ if ($ad_ergausg->AdrArt == 'privat')
     {
     if (($ad_ergausg->Strasse != '') and ($ad_ergausg->Ort !=''))
        {
        $vcard->strasse_privat = $ad_ergausg->Strasse;
        $vcard->plz_privat = $ad_ergausg->PLZ;
        $vcard->ort_privat = $ad_ergausg->Ort;
        }
❾ if ($ad_ergausg->AdrArt == 'dienst')
     {
```

```
     if (($ad_ergausg->Strasse != '') and ($ad_ergausg->Ort !=''))
       {
       $vcard->strasse_dienst = $ad_ergausg->Strasse;
       $vcard->plz_dienst = $ad_ergausg->PLZ;
       $vcard->ort_dienst = $ad_ergausg->Ort;
       }
     }


❿ $vcard->zusammensetzen();
  $vcard->speichern();

  $ausgabe = $vcard->ausgabe($aufrufurl);
  echo $ausgabe;
}
```

6.3.3.4 suche_lv.inc

suche_lv.inc enthält zwei Funktionen und eine Klasse: `lv()`, `sigel_aufbereit()` und `lv_detail` Die wichtigste Funktion ist `lv()`, sie überprüft, ob, wann und wo ein Dozent eine Lehrveranstaltung hält. `sigel_aufbereit()` ist nur eine Hilfsfunktion für `lv()`. Sie dient zur Aufbereitung eines SQL-Teilstrings. Ihr wird eine Ergebniskennung übergeben, holt die einzelnen Datensätze ab ❶. Ein Datensatz enthält je ein Sigel einer Lehrveranstaltungen. Die Sigel werden dann für eine weitere Suche mit OR verknüpft ❷.

```
function sigel_aufbereit($erg)
{
$sigel = "(";
for ($h=0; $h<mysql_num_rows($erg); $h++)
     {
❶    $erg_obj = mysql_fetch_object($erg);
     $sigel .= "Sigel = $erg_obj->Sigel";
❷    if ($h < (mysql_num_rows($erg)-1))
       $sigel .= " OR ";
     else
       $sigel .= ")";
     }
return $sigel;
}
```

Die Klasse `lv_detail` ist für die Speicherung der Eigenschaften einer gefundenen Veranstaltung. Diese Eigenschaften werden aus verschiedenen Datenbanken und Tabellen abgefragt und für die weitere Verarbeitung ist es einfacher, wenn auf sie in der Form `$objekt->eigenschaft` zugegriffen werden kann.

```
class lv_detail
{
```

```
var $typ;
var $titel;
var $beginn_stunde;
var $ende_minute;
var $ende_stunde;
var $ende_minute;
var $raum;

function lv_detail()  //Konstruktor
{
  $this->typ = "reg";
  $this->beginn_minute = "15";
  $this->ende_minute = "00";
}
}
```

Am Anfang von `lv()` stehen einige vorbereitende Anweisungen: aus dem übergebenen Timestamp wird für die Datenbankabfrage der Wochentag extrahiert ❶, für die Ausgabe werden Stunde und Datum formatiert ❷. Für die spätere Ergebnisausgabe wird der vollständige Name des Dozenten abgerufen und aufbereitet ❸.

```
function lv($dozent_einzel_nummer, $timestamp)
{
❶ switch(date("w", $timestamp))
  {
  case 0:
  $zeit[tag] = "so";
  break;
  ...                // gekürzt aus Platzgründen
  case 6;
  $zeit[tag] = "sa";
  break;
  }

❷ $zeit[stunde] = date("G", $timestamp);
  $zeit[datum] = date("Y-m-d", $timestamp);

❸ $abfr_zeit_verb = mysql_connect($db_config[host],  ⏎
                    $db_config[user], $db_config[passwd]);

  $erg_lv_dozent = mysql_db_query("ws2001", "SELECT Name, Vorname  ⏎
                   FROM dozadr WHERE nummer=$dozent_einzel_nummer");
  $ergausg_lv_dozent = mysql_fetch_object($erg_lv_dozent);
     include_once('hilf_funk.inc');
  $name=titel($ergausg_lv_dozent->Vorname, $ergausg_lv_dozent->Name);
```

Hier folgt dann die eigentliche Suche von Lehrveranstaltungen. Da der Quellcode sehr lang ist habe ich viele Anweisungsblöcke durch Kommentare ersetzt. So bleiben die eigentlich wichtigen Kontrollstrukturen ersichtlich.[17]

Zuerst werden alle regulären und einmaligen Lehrveranstaltungen(LV) des Dozenten abgefragt, dann wird überprüft, welche der Lehrveranstaltungen am richtigen Tag stattfinden. Es ist leider nicht möglich, dies mit einer SQL-Abfrage zu erledigen. Dazu wäre ein Sub-Select nötig, d. h. eine SELECT-Anweisung in einer anderen SELECT-Anweisung. mySQL unterstützt diese Art der Abfrage leider nicht. Ein weiterer Workaround befindet sich bei ❶: In der Tabelle „einmalig“ sind die Dozenten nicht mit ihren eindeutigen Nummern eingetragen, sondern nur mit ihrem Namen. Die Rechts- und Linkstrunkierung ist notwendig da im Feld „dozentname“ auch mehrere Namen stehen können (getrennt durch Schrägstrich, Komma o.ä.).

```
  // Array, speichert, ob Suchergebnis gefunden bzw. wieviele
$erg_num = array("reg" => -1, "einmalig" => -1);

  // Abfrage nach allen regulären LVs
$erg_reg_lv = mysql_db_query("ws2001", "SELECT DISTINCT Sigel  ⏎
              FROM dozenten WHERE Dozent=$dozent_einzel_nummer");
if (@mysql_num_rows($erg_reg_lv) > 0)
$erg_num[reg]++;

  // Abfrage nach allen einmaligen LVs
❶ $erg_einmalig_lv = mysql_db_query("stplankontakt", "SELECT Sigel  ⏎
                  FROM einmalig WHERE dozentname LIKE  ⏎
                  '%$ergausg_lv_dozent->Name%'");

if (@mysql_num_rows($erg_einmalig_lv) > 0)
  $erg_num[einmalig]++;

if ($erg_num[reg] == 0)
  $sigel_reg_lv = sigel_aufbereit($erg_reg_lv);
if ($erg_num[einmalig] == 0)
  $sigel_einmalig_lv = sigel_aufbereit($erg_einmalig_lv);

  // LVs, die an gewünschtem Tag sind, werden abgefragt
if (isset($sigel_reg_lv))
  {
  $erg_lv_reg_tag = mysql_db_query("ws2001", "SELECT  ⏎
                DISTINCT Sigel, Zeit  ⏎
                FROM lv  ⏎
                WHERE $sigel_reg_lv AND Tag LIKE '$zeit[tag]'  ⏎
```

[17] Der vollständig Quelltext, wie auch die Quelltexte aller anderen Funktionen und Klassen befinden sich auf dem beigelegten Datenträger.

```
                ORDER BY Zeit");
  $erg_num[reg] += @mysql_num_rows($erg_lv_reg_tag);
  }

if (isset($sigel_einmalig_lv))
  {
  $erg_lv_einmalig_tag = mysql_db_query("stplankontakt", "SELECT
                Sigel, zeit, veranstaltung, raum
                FROM einmalig
                WHERE ($sigel_einmalig_lv
                AND datum LIKE '$zeit[datum]')");
  $erg_num[einmalig] += @mysql_num_rows($erg_lv_einmalig_tag);
  }
```

Falls Lehrveranstaltungen für den passenden Tag gefunden wurden, wird ihre Dauer aus der Tabelle „dozenten“ abgefragt. Der Beginn und das errechnete Ende werden mit der gesuchten Uhrzeit verglichen ❶ - ❸. Je nachdem, wie die Veranstaltung im Verhältnis zur gesuchten Stunde liegt, werden ihre Eigenschaften (Titel, Raum, Zeit, etc.) in den Objekten `$vor`, `$jetzt` oder `$nach` gespeichert.

```
  // Es gibt LVs an diesem Tag
if (($erg_num[reg] > 0) or ($erg_num[einmalig] > 0))
{
  if ($erg_num[reg] > 0)
  {
  for ($h=0; $h<$erg_num[reg]; $h++)
    {
        // Für jede LV wird die Dauer abgefragt
        // Stunde wird umgerechnet
    $ergausg_lv_sigel_zeit->Zeit += 8;

        // Vergleich von LV-Zeit und gewünschtem Zeitpunkt
        // LV endet vor gewünschtem Zeitpunkt
❶   if (($ergausg_lv_sigel_zeit->Zeit + $ergausg_lv_dauer) <
                                              $zeit[stunde])
      {
            // Eigenschaften der LV werden abgerufen und
            // in Objekt $vor gespeichert
      }
        // LV läuft zu gewünschtem Zeitpunkt
❷   elseif (($ergausg_lv_sigel_zeit->Zeit <= $zeit[stunde]) and
     ($stunde <= ($ergausg_lv_sigel_zeit->Zeit + $ergausg_lv_dauer)))
      {
            // Eigenschaften der LV werden abgerufen und
            // in Objekt $jetzt gespeichert
      }
```

```
        // LV beginnt nach gewünschtem Zeitpunkt
❸   else
      {
          // Nur die direkt nachfolgende LV soll berücksichtigt
          // werden, da LVs in Abfrage nach Anfangszeit geordnet
          // waren ist erste 'Nach-LV' die am nächsten darauffolgende
        if (empty($nach))
        {
          // Eigenschaften der LV werden abgerufen und
          // in Objekt $nach gespeichert
        }
      }
    }
  }
```

Falls einmalige Lehrveranstaltungen gefunden wurden, folgen fast dieselben Anweisungen, mit einigen kleine Unterschieden: Der Beginn und das Ende der Lehrveranstaltung sind in der Datenbank in einem String gespeichert. Die einzelnen Stunden- und Minutenangaben müssen extrahiert werden ❶. Beginn und Ende der einmaligen Vorlesung wird nicht nur mit der gesuchten Uhrzeit verglichen, sondern auch mit einer eventuell ebenfalls vor/nach/zu dieser Uhrzeit liegenden regulären Veranstaltung verglichen ❷. Liegt eine einmalige und eine reguläre Veranstaltung parallel, wird der regulären der Vorzug gegeben.

```
if ($erg_num[einmalig] > 0)
  {
  for ($i=0; $i<$erg_num[einmalig]; $i++)
  {
  $ergausg_lv_einmalig = mysql_fetch_object($erg_lv_einmalig_tag);

❶ list($beginn_stunde_einmalig, $beginn_minute_einmalig, ⏎
      $ende_stunde_einmalig, $ende_minute_einmalig) =  ⏎
      split('[:-]', $ergausg_lv_einmalig->zeit);

  if ((($beginn_stunde_einmalig < $zeit[stunde])  ⏎
    and ($ende_stunde_einmalig < $zeit[stunde]))  ⏎
❷   and (!isset($vor)  ⏎
    or (isset($vor) and ($vor->ende_stunde<$beginn_stunde_einmalig))))
    {
              // Eigenschaften der LV werden abgerufen und
              // in Objekt $vor gespeichert
    }

  elseif ((($beginn_stunde_einmalig <= $zeit[stunde])  ⏎
    and ($ende_stunde_einmalig >= $zeit[stunde])) and !isset($jetzt))
    {
```

```
                // Eigenschaften der LV werden abgerufen und
                // in Objekt $jetzt gespeichert
    }

  elseif (($beginn_stunde_einmalig > $zeit[stunde])  ⏎
    and (!isset($nach)  ⏎
    or (isset($nach)  ⏎
    and $nach->beginn_stunde > $beginn_stunde_einmalig)))
    {
                // Eigenschaften der LV werden abgerufen und
                // in Objekt $nach gespeichert
    }
  }
}
```

Ab hier muß nicht mehr zwischen einmaligen und regulären Veranstaltungen unterschieden werden. Die vorher, nachher oder gerade stattfindenden Vorlesungen sind in `$vor`, `$nach` bzw. `$jetzt` gespeichert, egal ob sie regulär oder einmalig sind.

Für die Ausgabe werden die in `$vor`, `$nach` bzw. `$jetzt` gespeicherten Daten zu einem kleinen Auskunftstext zusammengesetzt. Dabei wird zwischen vier Fällen unterschieden: Die Lehrveranstaltung läuft gerade ❶, die letzte Veranstaltung des Tages ist vorbei ❷, eine Veranstaltung ist vorbei, aber es folgt noch eine weitere ❸ oder die Veranstaltung beginnt erst noch ❹. Je nachdem, welche Daten über die Veranstaltung vorhanden sind (z. B. fehlt teilweise eine Raumangabe), wird der Ausgabetext verändert ❺.

```
❶ if (isset($jetzt))
  {
  $ausgabe = "<i>$name[gesamt]</i> hält im Moment die  ⏎
              Lehrveranstaltung \"$jetzt->titel\"";
  ❺ if (!empty($jetzt->raum))
      $ausgabe.= " in Raum $jetzt->raum";
  $ausgabe.= ". Die Lehrveranstaltung endet voraussichtlich um  ⏎
             $jetzt->ende_stunde:$jetzt->ende_minute Uhr.  ⏎
             ($zeit[datum])";
  }
❷ elseif (isset($vor) and !isset($nach))
  {
  $ausgabe = "Die letzte Lehrveranstaltung von  ⏎
             <i>$name[gesamt]</i> endet um  ⏎
             $vor->ende_stunde:$vor->ende_minute Uhr";
    if (!empty($vor->raum))
     $ausgabe .= " in Raum $vor->raum";
  $ausgabe .= ". Es sind keine weiteren Veranstaltungen für  ⏎
              diesen Tag eingetragen. ($zeit[datum])";
  }
```

```
❸ elseif (isset($vor) and isset($nach))
  {
  $ausgabe = "Um $vor->ende_stunde:$vor->ende_minute Uhr endet  ⏎
              die vorhergehende Lehrveranstaltung von  ⏎
              <i>$name[gesamt]</i>";
    if (! empty($vor->raum))
     $ausgabe.= " in Raum $vor->raum";
  $ausgabe.= ", um $nach->beginn_stunde:$nach->beginn_minute Uhr  ⏎
                beginnt die nächste Veranstaltung";
    if (! empty($nach->raum))
     $ausgabe.= " in Raum $nach->Raum";
  $ausgabe.= ".($zeit[datum])";
  }

❹ elseif (!isset($vor) and isset($nach))
  {
  $ausgabe = "Um $nach->beginn_stunde:$nach->beginn_minute Uhr  ⏎
              beginnt die erste Lehrveranstaltung von  ⏎
              <i>$name[gesamt]</i> für diesen Tag";
    if(! empty($nach->raum))
     $ausgabe.= " in Raum $nach->raum";
  $ausgabe.= ". ($zeit[datum])";
  }
}
```

Für den Fall, daß keine Lehrveranstaltungen gefunden werden (am gesuchten Tag bzw. überhaupt), werden natürlich auch Sätze für die Ausgabe gebildet.

```
  // Es gibt keine LVs an diesem Tag
elseif (($erg_num[reg] == 0) and ($erg_num[reg] == 0))
  $ausgabe = "<i>$name[gesamt]</i> hat an diesem Tag  ⏎
              keine Lehrveranstaltungen. ($zeit[datum])";

  //Es wurden gar keine LVs fuer den Dozenten gefunden
elseif (($erg_num[reg] == -1) and ($erg_num[einmalig] == -1))
  $ausgabe = "<i>$name[gesamt]</i> hält dieses Semester  ⏎
              keine Lehrveranstaltungen.";

elseif (!isset($erg_num))
$ausgabe = "Leider ist ein Fehler aufgetreten.";
```

Um den Quellcode lesbarer zu halten, werden erst bei der Rückgabe der Erreichbarkeitsinformation Umlaute durch Entitäten ersetzt.

```
return umlaute($ausgabe);
}
```

6.3.4 Fehlerbehandlung

In den erläuterten Quelltextbeispielen wurden die Anweisungen für die Fehlerbehandlung entfernt, im Originalquelltext sind sie aber vorhanden.

Bei Ausdrücken, die einen Fehler verursachen können, wird die Ausgabe der von PHP erzeugten Fehlermeldung durch den Operator `@` unterdrückt. Die Ausführung des Skriptes wird mit einer selbst definierten Meldung abgebrochen. Je nachdem, in welchem Verhältnis die Ausgabe einer Funktion zu einem WML-Dokument steht, muß die Fehlermeldung WML-Tags enthalten (z. B. um das Dokument oder die Card zu beenden).

Beispiele für den Einsatz der Fehlerbehandlung sind:

- Verbindungsaufbau zu mySQL

```
$verbindung = @mysql_connect(...)
 or die ("<card><p>Im Moment ist leider keine  ↵
          Datenbank-Verbindung m&#246;glich.</p></card></wml>");
```

- Auswahl einer Datenbank

```
$db = @mysql_select_db(...)
or die ("Eine n&#246;tige Datenbankabfrage kann leider nicht  ↵
          durchgef&#252;hrt werden.");
```

- Abfrage einer Datenbank

```
$erg = @mysql_db_query(...)
or die ("<card><p>Eine n&#246;tige Datenbankabfrage kann  ↵
          leider nicht durchgef&#252;hrt werden.</p></card></wml>");
```

- Ermitteln der Ergebnisanzahl

```
$erg_anzahl = @mysql_num_rows(...);
```

  Falls die Suche nicht erfolgreich war, kommt es z. T. zu Fehlermeldungen, wenn man die Anzahl der Ergebnisse ermitteln will. Es genügt, die Ausgabe dieser Meldung zu unterdrücken, da bei keinem Ergebnis `$erg_anzahl` richtigerweise auf 0 gesetzt wurde.

- Verzeichnis-/Dateioperationen

```
$verzeichnis = @opendir("vcf");
$vcf = @fopen($dateiname, "w");
@fwrite($vcf, $this->vcard);
@fclose($vcf);
@chmod($dateiname, 0755);
@closedir($verzeichnis);
```

  Hier habe ich auf die Ausgabe einer Fehlermeldung verzichtet, da die Anweisungen in der Methode einer Klasse stehen, an einer Stelle, an der ein Abbruch der Skriptausführung den Gesamtablauf empfindlich stören würde.

6.4 Ergänzung der Stundenplan-Datenbank

In der bestehenden Stundenplan-Datenbank fehlen einige Angaben, die für Benutzer der Erreichbarkeitsauskunft bzw. des Verzeichnisdienstes interessant sein könnten: Einmalige Lehrveranstaltungen, Abwesenheitszeiten, Büronummern.

Um diese Daten aufzunehmen habe ich die Datenbank „stplankontakt“ erstellt. Die Tabelle „personen“ hatte ich als Dump vorliegen, konnte ich also über den Kommandozeilenbefehl `mysql stplankontakt < personen.sql` importieren.

Die einmaligen Veranstaltungen liegen auf dem Lehre-Server nur als eine Tabelle im HTML-Format vor. Mit einigen Suchen-und-Ersetzen Operationen in einer Textverarbeitung habe ich aus dem HTML-Quellcode der Datei eine Textdatei gemacht, die über den Kommandozeilenbefehl `mysqlimport stplankontakt einmalig.txt` importiert werden konnte. Voraussetzung hierfür ist, daß pro Zeile ein Datensatz steht, dessen Felder durch Tabulatoren getrennt sind. Vorher hatte ich noch die entsprechende Tabelle „einmalig“ erstellt mit folgenden Feldern und Datentypen:

- `Sigel, int(11), auto_increment, primary key`
- `datum, date`
- `tag, char(2)`
- `raum, varchar(5)`
- `zeit, varchar (20)` : Datentyp time war nicht möglich, da das Format der Zeitangaben „hh:mm-hh:mm ist.“
- `dozentname, varchar(50)`
- `veranstaltung, varchar(255)`

6.5 Probleme

Die meisten Probleme bei der Programmierung bereitete die Datengrundlage. Es waren einige fehlerhafte Eintragungen vorhanden, so als gäbe es keine Kontrolle bei der Eingabe neuer Datensätze. Da ich nur Lesezugriff auf die Datenbank habe und diese Fehler nicht einfach in der Datenbank korrigieren konnte, waren einige Anweisungen nötig, die im Nachhinein die aus der Datenbank abgerufenen Daten überprüfen und gegebenenfalls richtigstellen.

Beispiel dafür ist ein Teil der Funktion `wtai()`:

```
$tel_nummer = ereg_replace("^[^0-9]*|[^0-9]*$", "", $tel_nummer);
```

Hier werden vor oder nach der Telefonnummer stehende nicht-numerisch Zeichen entfernt: aus „Tel.: 0711/1234567“ wird „0711/1234567“, aus „0711/1234567
“ wird „0711/1234567“. Das Ärgerliche ist, daß manchmal nur ein einziger Datensatz wirklich betroffen ist, und alle anderen durch die Anweisung unberührt bleiben.

Ähnlich verhält es sich mit den Angaben zu Titel, Vorname und Nachname des Dozenten. Statt den Titel in einem eigenen Feld zu speichern, steht er hinter den Vornamen: „Christian, Dr.“. Würde einfach Vorname Nachname ausgegeben, ergibt sich die inakzeptable Ausgabe „Christian, Dr. Rahtke“. Also muß der Titel vom Vornamen getrennt und in einer eigene Variablen gespeichert werden. Zuständig ist ein Teil der Funktion `titel()`:

```
if (ereg(",", $vornametitel))
list($vorname, $titel) = explode(",", $vornametitel);
elseif (ereg(" ", $vornametitel))
list($vorname, $titel) = explode(" ", $vornametitel);
else
$vorname = $vornametitel;
```

Das Ärgerliche hier war, daß in den meisten Fällen der Titel mit einem Komma vom Vornamen getrennt war. Aber eben nur in den meisten Fällen. Bei einigen wenigen (genauer: 4) fehlte das Komma, weswegen eine Fallunterscheidung nötig war.

Das zweitgrößte Problem war die Verteilung der Daten auf die Tabellen der Datenbanken. Bei der Suche nach Lehrveranstaltungen eines Dozenten an einem bestimmten Tag muß diese Abfrage in mehreren Schritten erfolgen, da mySQL den nötigen Sub-Select nicht unterstützt. Da die Datengrundlage zusammengestückelt ist aus Stundenplan-Datenbank und ergänzenden Tabellen anderen Ursprungs, konnten Verknüpfungen nicht über Schlüsselfelder erfolgen, sondern über eine Freitextsuche in Nachname bzw. E-Mail-Adresse.

7 Ergebnis

Die Erreichbarkeitsauskunft und der Verzeichnisdienst sind voll funktionsfähig, beide erfüllen alle gestellten Anforderungen.

Anhand von Bildschirmausdrucken des Siemens S45-Simulators aus dem Openwave SDK will ich hier den Ablauf einer Benutzeranfrage demonstrieren. Der Zeitindex soll einen Eindruck verschaffen, wie schnell bzw. langsam die Abfrage aus dem Mobiltelefonnetz ist. Die Zeiten wurden ermittelt im T-D1-Netz von T-Mobile. Die Übertragungsgeschwindigkeit waren 9600 Bit/s, das Endgerät ein Siemens S35i mit dem UP.Browser, Version 4.1.16.

Tabelle 17: Benutzeranfrage über den Siemens S45-Simulator

Zeitindex	**Siemens S45-Simulator**	**Aktion**
0 s		Der Benutzer wählt die Seite in seinen Lesezeichen aus, das Telefon stellt die Verbindung her.
16 s		Die Begrüßungsseite wird angezeigt. Falls der Benutzer nicht „Weiter“ aktiviert, wird er nach 3 Sekunden weitergeleitet.
20 s		Er bekommt eine kurze Erklärung angezeigt, wie die Auskunft funktioniert.

24 s		Jetzt kann er einen Namensanfang eintragen und den „Weiter“-Link aktivieren.
30 s	  	Er bekommt eine Auswahl angeboten, für welchen Zeitpunkt er Erreichbarkeitsinformationen der gewünschten Person einholen will.
38 s	  	Der Benutzer entscheidet sich dafür, den genauen Zeitpunkt einzugeben. Er gibt den Tag ein, wählt den Monat aus und bestimmt die Uhrzeit. Jetzt werden die Formulardaten an den Server geschickt.

43 s	 	Die Erreichbarkeitsinformationen werden angezeigt.
51 s	   	Der Benutzer lässt sich die Kontaktdaten anzeigen.
54 s		Wenn er eine Telefonnummer auswählt, kann er sie wählen oder abspeichern lassen.
57 s	 	Falls es sein Telefon unterstützt, kann er die Kontaktdaten in Form einer vCard abspeichern.

In weniger als einer Minute sind alle Informationen abgefragt!

8 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurden zwei WAP-Services für die Hochschule der Medien entwickelt, eine Erreichbarkeitsauskunft und ein Verzeichnisdienst. Dozenten und Studenten haben nun die Möglichkeit, wohin auch immer sie ihr Mobiltelefon mitnehmen, Adressen, Telefonnummern und andere Auskünfte abzufragen.

Diese Arbeit hat mir gezeigt, daß es einen großen Unterschied gibt, zwischen der Entwicklung von Web- und WAP-Angeboten. Bei der Entwicklung für das Wireless Application Protocol gilt es, sich auf das Wichtigste zu beschränken. Design als Selbstzweck, wie wir es von Webseiten kennen, wird zweitrangig. Doch gerade wenn ein WAP-Angebot schlecht designt ist, wird es keinen Nutzen für seine User bringen. Es wird sie eher von der Nutzung weiterer WAP-Dienste abhalten.

Ich hoffe, daß die WAP-Services der HdM nicht zu dieser Gruppe gehören werden. Es war mir ein Anliegen, das Angebot für den Nutzer so zu gestalten, daß er die Möglichkeiten der mobilen Information gezielt einsetzen kann.

In Zukunft, könnte ich mir vorstellen, daß die WAP-Services der Hochschule noch weiter ausgebaut werden. Beispielsweise könnte es einen Studenten-Login geben, unter dem jeder Studierende seine Hochschul-E-Mail-Adresse auf neue Post überprüfen oder die Termine seiner nächsten und die Noten seiner letzten Klausuren einsehen kann, egal, wo er sich gerade befindet.

Literaturverzeichnis

Heijden, M. van der, Frost, M. (2000): The Wireless Application Environment for Creating WAP Services and Applications. In: Heijden, M. van der, Frost, M. (Hrsg.): Understanding WAP. Wireless Applications, Devices and Services. Artech House, Norwood, MA.

Immler, C., Kreinacke, M., Spallek, A. (2000): WAP. Das große Buch. DATA BECKER, Düsseldorf.

Schmid, E. (Hrsg.) (2001): PHP-Handbuch. http://www.php.net/docs.php. (Datum des Zugriffs: 07.08.2001).

Seeboerger-Weichselbaum, M. (2000): WAP-Programmierung mit WML und WMLScript. bhv Verlag, Kaarst.

Taylor, M., Hosking, I, Brazier, D. (2000): Designing Effective User Interfaces for WAP Services. In: In: Heijden, M. van der, Frost, M. (Hrsg.): Understanding WAP. Wireless Applications, Devices and Services. Artech House, Norwood, MA.

versit Consortium (1996): vCard. The Electronic Business Card. Version 2.1. A versit Consortium Specification. http://www.imc.org/pdi/vcard-21.txt. (Datum des Zugriffs: 11.10.2001).

WAP Forum (2001a): WAP Architecture. Version 12-July-2001. Wireless Application Protocol Architecture Specification. WAP-210-WAPArch-20010712. Wireless Application Protocol Forum, Ltd. http://www.wapforum.org[18].

WAP Forum (2001b): Wireless Markup Language. Version 2.0. Proposed Version 26-June-2001. Wireless Application Protocol. WAP-238-WML-20010626-p. Wireless Application Protocol Forum, Ltd. http://www.wapforum.org.

WAP Forum (2001c): WAP Wireless Telephony Application. Proposed Version 11-JUL-2001. Wireless Application Protocol. WAP-266-WTA-20010711-p. Wireless Application Protocol Forum, Ltd. http://www.wapforum.org.

WAP Forum (2001d): WAP Wireless Telephony Application Interface. Proposed Version 15-Jul-2001. Wireless Application Protocol. WAP-268-WTAI-20010715-p. Wireless Application Protocol Forum, Ltd. http://www.wapforum.org.

WAP Forum (2001e): Binary XML Content Format Specification Version 1.3, 25 July 2001. Wireless Application Protocol. WAP-192-WBXML-20010725-a. Wireless Application Protocol Forum, Ltd. http://www.wapforum.org.

[18] Leider kann für die Dokumente des WAP Forum keine genau URL angegeben werden, da man sie erst nach einem vorgeschalteten Copyright-Hinweis betrachten und abspeichern kann.

WAP Forum (2001f): WMLScript Crypto Library. Version 20-Jun-2001. Wireless Application Protocol. WAP-161-WMLScriptCrypto-20010620-a. Wireless Application Protocol Forum, Ltd. http://www.wapforum.org.

WAP Forum (2001g): Wireless Application Environment Defined Media Type Specification. Proposed Version 15-May-2001. Wireless Application Protocol. WAP-237-WAEMT-20010515-p. Wireless Application Protocol Forum, Ltd. http://www.wapforum.org.

WAP Forum (2000a): WMLScript Specification. Version 25-Oct-2000. WAP-193-WMLS-20001025-a. Wireless Application Protocol Forum, Ltd. http://www.wapforum.org.

WAP Forum (2000b): WMLScript Standard Libraries Specification. 25-SEP-2000. Wireless Application Protocol. WAP-194-WMLSL-20000925-a. Wireless Application Protocol Forum, Ltd. http://www.wapforum.org.

Wenz, C., Hauser, T. (2001): WAP. Architektur - Programmierung - Referenz. 1. Auflage. Carl Hanser Verlag, München - Wien.

Erklärung

Hiermit erkläre ich, daß ich die vorliegende Diplomarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ort, Datum	Unterschrift